application note

# SC1000-Family Program-Memory Bus Activity

StarCore LLC
8303 Mopac Expy, Ste A400
Austin, TX 78759, USA
Phone: +1-512-682-8500
Fax: +1-512-682-8699
Email: support@starcore-dsp.com
Web: www.starcore-dsp.com

Document number 10014-01 (2 April 2003)

StarCore
Brighter DSP Technology

# Table of Contents

# Chapter 1: Introduction

## Overview

**Purpose**

This application note explains how the StarCore™ SC1000-family DSP cores use the program-memory address and data buses and a five-stage pipeline to fetch and dispatch instructions. This note describes the behavior of the cores from a programming-model perspective only; it doesn't detail the inner workings of the pipeline. Using the information in this note, you can improve assembly-code performance, reduce power consumption, and further decrease the likelihood of memory-access conflicts.

**Primary audience**

This application note is for individuals who write assembly-language programs for the SC1000-family DSP cores.

**For more information**

For more information on the SC1000-family core pipeline, as well as short and long hardware loops, see the *SC1200 DSP Core Reference Manual* (document number 10000) or the *SC1400 DSP Core Reference Manual* (document number 10001).

# Chapter 2: How the PSEQ Works

## Overview

**Introduction**

This chapter introduces several important terms and describes the process the SC1000-family DSP cores use to fetch and dispatch instructions.

**Variable-length execution sets**

On the StarCore DSP cores, a group of instructions that execute simultaneously is called a *variable-length execution set*, or *VLES*. In assembly code, a VLES appears as either several instructions on the same line or a group of instructions enclosed in brackets.

**Pipeline stages**

The SC1000-family DSP cores process VLESes using a five-stage pipeline:

| Stage | Name |
|---|---|
| 1 | Prefetch |
| 2 | Fetch |
| 3 | Dispatch |
| 4 | Address generation |
| 5 | Execution |

**Program sequencer unit**

The *program sequencer unit (PSEQ)* is the logical unit that handles the prefetch, fetch, and dispatch stages.

**In this chapter**

This chapter covers the following topics:

| Topic | Page |
|---|---|
| Fetch Sets | 8 |
| Fetch Buffers | 9 |
| PSEQ Behavioral Example | 11 |

# Fetch Sets

**Fetch sets**

A *fetch set* is a series of instruction words in program memory that

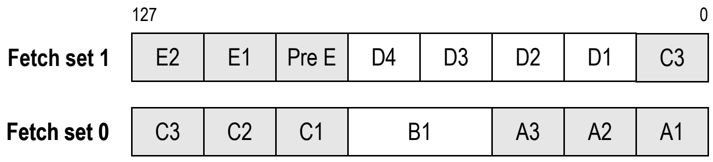
- The DSP core reads from the program-memory data bus (PDB) during the fetch stage
- Is 128 bits (eight 16-bit words) long
- Is naturally aligned on an eight-word boundary

**VLES alignment in fetch sets**

Because the SC1000 architecture doesn't restrict the alignment of the start of a VLES, a VLES can start at any word in a fetch set and extend over the eight-word fetch-set boundary.

**Example: Fetch sets**

This is an example of two fetch sets in memory that contain five VLESes, labeled A, B, C, D, and E:

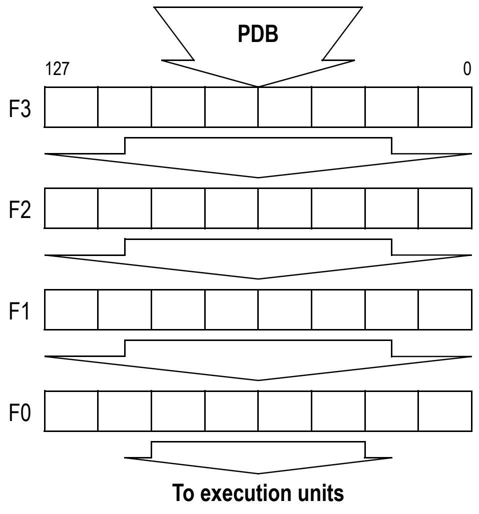


VLES A contains three 16-bit instructions. VLES B contains a single 32-bit instruction. VLES C contains three instructions—two 16-bit instructions and one 32-bit instruction that crosses a fetch-set boundary. VLES D contains four 16-bit instructions. VLES E contains two 16-bit instructions and a 16-bit prefix (Pre E).

## Fetch Buffers

**Fetch buffers**

To ensure that the PSEQ dispatches instructions in a VLES with minimal timing penalties, the fetch stage contains four 128-bit fetch buffers: a primary buffer, F0, and three secondary buffers, F1 through F3.



**Buffer and VLES characteristics**

To understand how the PSEQ performs dispatching, it's helpful to keep the following characteristics of fetch buffers and VLESes in mind:

- The start of the current VLES used in the dispatch stage is always located in the primary buffer.
- Because the instruction-grouping rules limit a VLES to eight words, a VLES that extends beyond the primary buffer never extends beyond the secondary buffer F1.

**How dispatching works**

This table explains what the PSEQ does in the dispatch stage under various conditions:

| When | Then the PSEQ |
|---|---|
| A VLES that's being dispatched extends beyond the primary buffer | Continues to parse instruction words using secondary buffer F1 |
| A VLES that's being dispatched ends on the last word of the primary buffer or extends beyond the end of the primary buffer | Marks the primary buffer's contents as *invalid* |
| The contents of the primary buffer are invalid, or a VLES extends beyond the primary buffer but the secondary buffer F1 is invalid (as in the case of a change-of-flow to a VLES that spans the fetch-set boundary) | Suspends activity in the dispatch stage until the contents of the required buffers become valid |

## How the PSEQ resolves invalid fetch buffers

The PSEQ resolves invalid fetch buffers by transferring data in order from valid higher-numbered fetch buffers to lower-numbered invalid buffers and requesting the next fetch buffer in the program flow from memory. The PSEQ issues the address of the next fetch set to the program-memory address bus in the prefetch stage. In the next cycle, the fetch stage captures the resulting data on the program-memory data bus and routes it to the lowest-numbered invalid fetch buffer.

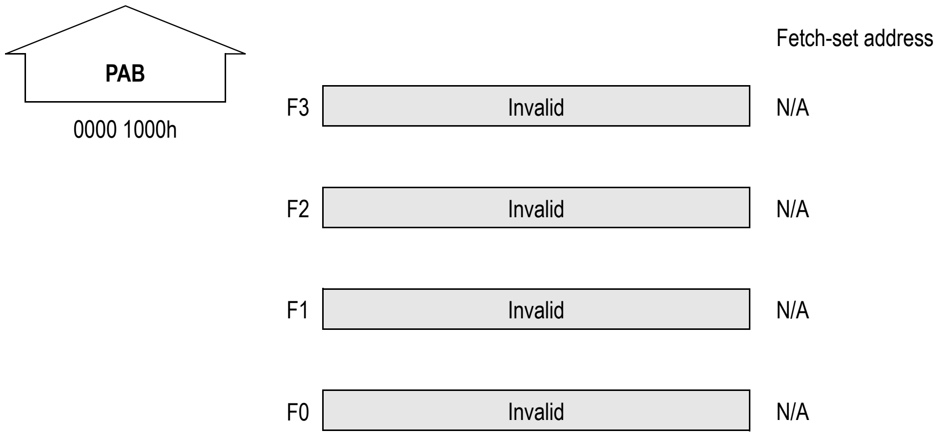
# PSEQ Behavioral Example

**Introduction**

The multicycle example in this section illustrates the behavior of the PSEQ with regard to the program-memory address bus (PAB), program-memory data bus (PDB), fetch buffers, and dispatcher. This example assumes that all of the fetch buffers are invalid at the start (a valid assumption assuming that a change-of-flow instruction has just executed). In this example, a VLES is represented by the letter *V* and a number (for example, V3).

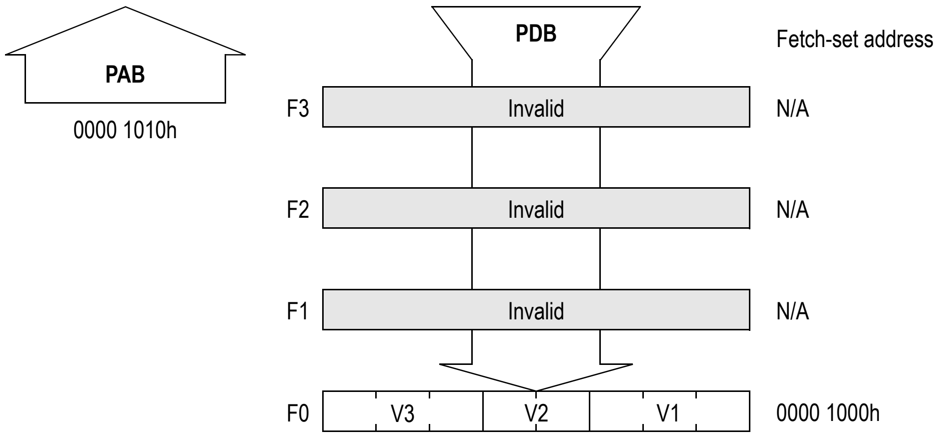
**Cycle N**

In cycle N, all fetch buffers are invalid. To resolve the invalid state, the prefetch stage issues a fetch-set address to the PAB. The address is for the fetch set that contains the next VLES to be executed. The PSEQ suspends activity in the fetch and dispatch pipeline stages until valid data is available in the fetch buffers.



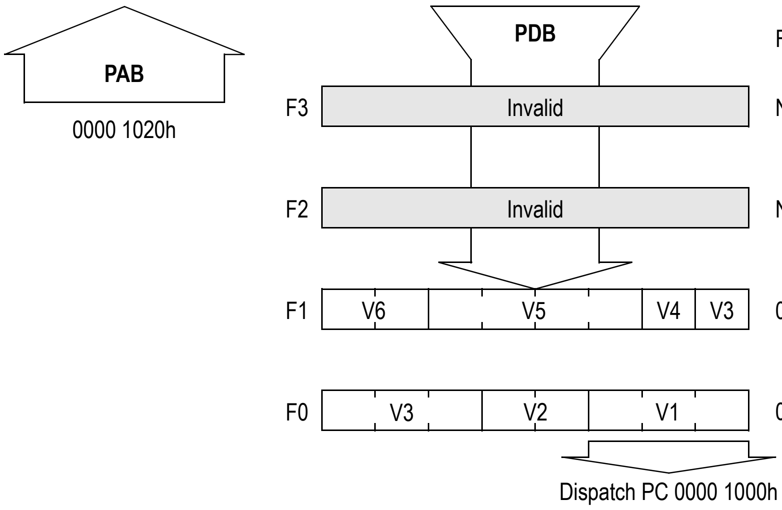
**Cycle N + 1**

In cycle N + 1, the PDB receives the program data for the fetch set requested in cycle N. The fetch stage routes this data to the lowest-numbered invalid fetch buffer—in this case, F0, the primary buffer. The other three fetch buffers are still invalid, so the prefetch stage issues the address of the next fetch set to the PAB. The dispatch stage remains suspended while the primary buffer is loaded from the PDB.



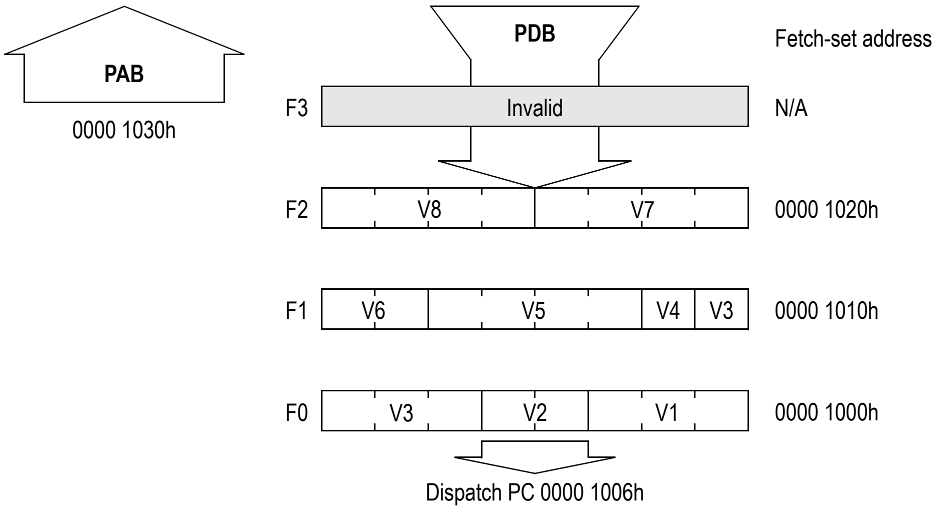
**Cycle N + 2**

In cycle N + 2, the primary buffer now contains a valid fetch set. The dispatch stage extracts the VLES starting at the program counter, or PC, (V1) and begins the process of dispatching for execution. Because the next VLES (V2) starts in the primary buffer, the buffer is still considered valid. The fetch stage routes the incoming data from the PDB to the lowest invalid fetch buffer (F1). The prefetch stage detects that invalid fetch buffers remain and issues the next fetch-set address to the PAB.



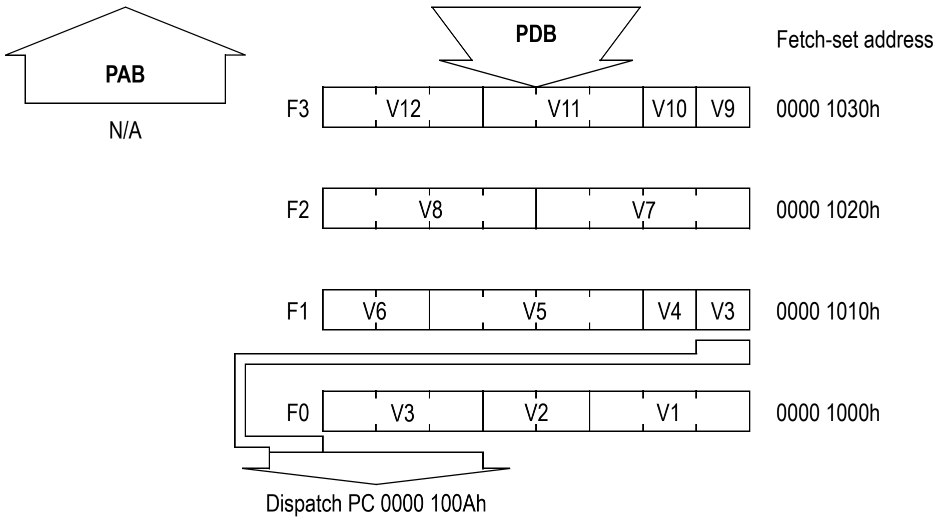
**Cycle N + 3**

In cycle N + 3, the dispatcher extracts VLES V2. Because the next VLES (V3) starts in the primary buffer, the buffer is still considered valid. The fetch stage routes the incoming data from the PDB to the lowest invalid fetch buffer (F2). The prefetch stage detects that a fetch buffer is invalid and issues the next fetch-set address to the PAB.



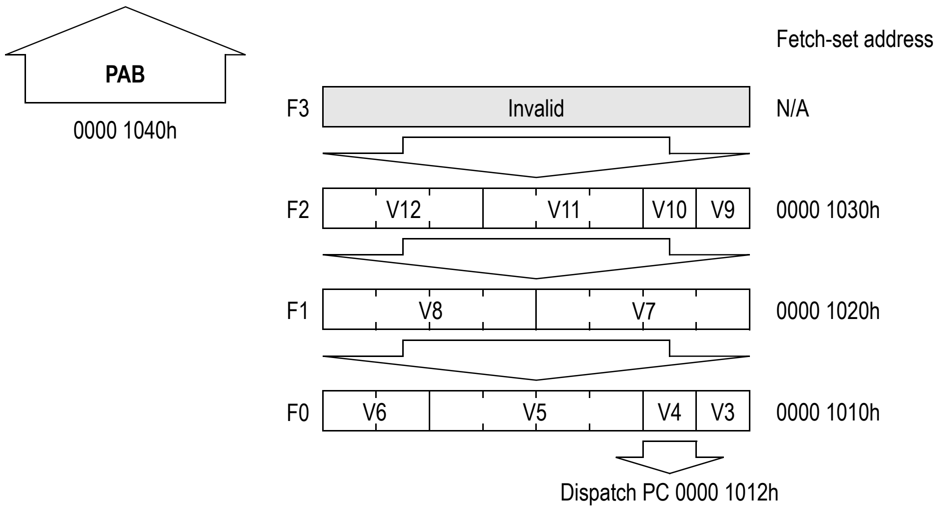
**Cycle N + 4**

In cycle N + 4, the dispatcher extracts VLES V3. V3 starts in the primary buffer and finishes in secondary buffer F1. The fetch stage routes the incoming data from the PDB to the lowest invalid fetch buffer (F3). Because the next VLES starts in a secondary buffer, the fetch stage also marks the primary buffer as invalid. With no secondary fetch buffers remaining to fill, the prefetch stage has nothing to do.



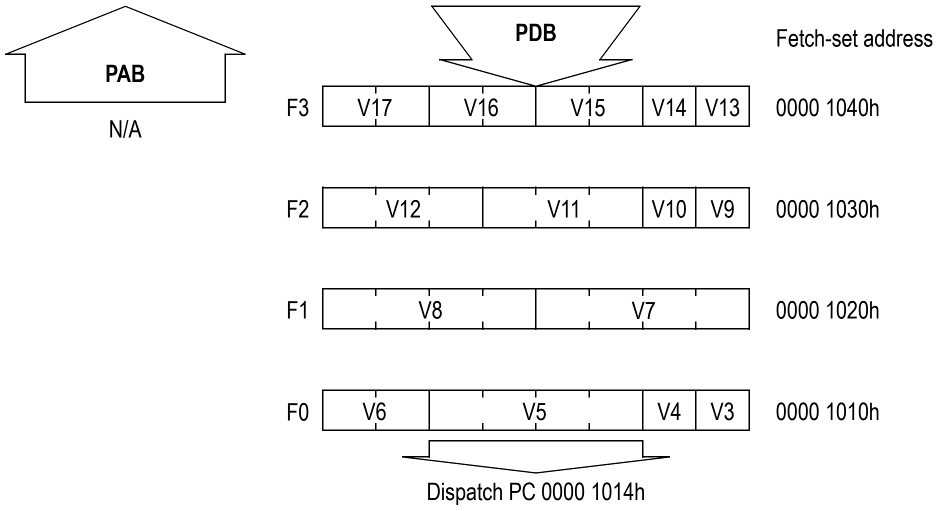
**Cycle N + 5**

In cycle N + 5, the PSEQ resolves the invalid state of the primary buffer by moving the contents of secondary buffer F1 into F0, the contents of F2 into F1, and the contents of F3 into F2. This leaves secondary buffer F3 invalid. The dispatcher extracts VLES V4 from the primary buffer. No program data was requested in the previous cycle, so there is no data to route from the PDB. Because the next VLES starts in the primary buffer, the primary buffer remains valid. The prefetch stage detects that a fetch buffer is invalid and issues the address of the next fetch set to the PAB.



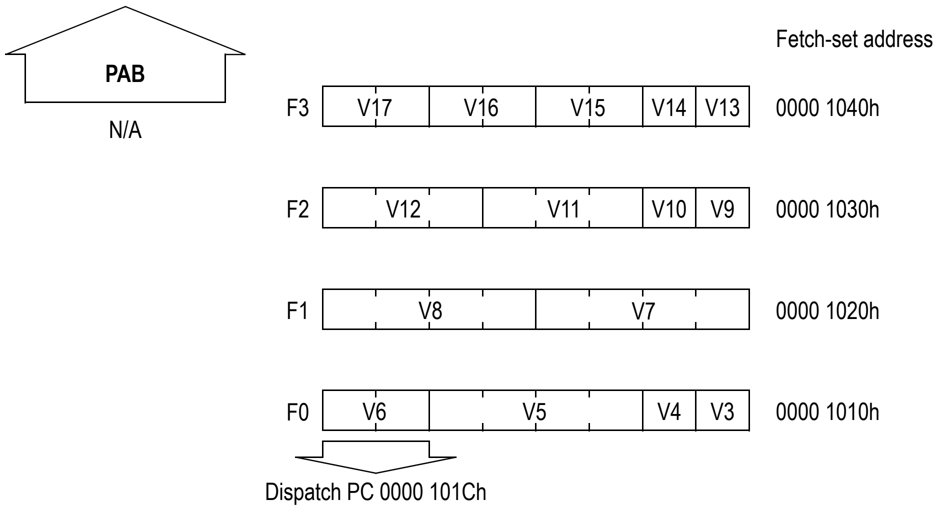
**Cycle N + 6**

In cycle N + 6, the dispatcher extracts VLES V5. The primary buffer remains valid. The PSEQ routes incoming data from the PDB to secondary buffer F3. No buffers are now invalid, so the prefetch stage is idle.



**Cycle N + 7**

In cycle N + 7, the dispatcher extracts VLES V6. The next VLES (V7) starts in secondary buffer F1, so the PSEQ marks the primary buffer as invalid. There is no incoming data to route from the PDB. The prefetch stage is idle because no buffer is invalid.



# Chapter 3: Examples—General Sequential Flow

## Overview

**Introduction**

This chapter and the next three chapters provide examples of program-memory bus activity based on the execution of actual code. These examples cover the corner cases and allow you to extrapolate other cases. For information on how to interpret these examples, see "About the Examples" on page 16.

This chapter gives examples of sequential program flow.

**What to notice**

The examples in this chapter illustrate the following points:

- Because code isn't typically composed of only eight-word VLESes, every cycle doesn't necessarily require a program-memory access. If the code is composed of only single-word VLESes, the core accesses program memory only once every eight cycles.
- Because activity on the program-memory buses is typically a couple of VLESes ahead of the current execution point, you can't use program-memory bus activity to isolate real-time debugging information without some postprocessing.

**In this chapter**

This chapter covers the following topics:

| Topic | Page |
|---|---|
| About the Examples | 16 |
| One-Word VLESes | 17 |
| Eight-Word VLESes That Are Fetch-Set Aligned | 19 |
| Eight-Word VLESes That Are Fetch-Set Unaligned | 21 |

# About the Examples

**Introduction**

This section provides important information about the examples in this chapter and the next three chapters.

**Code**

Each example begins with a code listing. A comment next to each VLES provides a unique VLES reference number and the program-memory address at which the VLES is located:

```
move.w #01,d1    ; V1, address 1000h
```

**Bus activity**

Each example ends with a table. Each row of the table shows the program-memory bus activity produced by the code in a given clock cycle, as well as the value of the program counter (PC) and the stage of execution of each VLES that's in the pipeline.

|  | 1 | 2 |  |  |  |  |  |  |  | 3 |  | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB |  |  |  |  |  |  |  | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – |  |  |  |  |
| 1 | 1010h | V8 | V7 | V6 | V5 | V4 | V3 | V2 | V1 |  |  |  |  |
| 2 | 1020h | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V10 | V9 | 1000h | V1 |  |  |

The numbers above the column headings indicate the associated pipeline stage, as defined in "Pipeline stages" on page 7.

**Bus-activity columns**

This table describes the information in each column of the bus-activity tables:

| Column | Description |
|---|---|
| PAB | The address present on the program-memory address bus (prefetch stage) |
| PDB | The data (VLESes) present on the program-memory data bus (fetch stage). A series of dashes indicates that there is no bus activity. Alternating gray and white areas represent separate VLESes. Dark gray areas represent valid instruction words that are defined outside the scope of the example code listing. |
| PC | The value of the program counter (dispatch stage) |
| VLES | The VLES being handled by the indicated stage (dispatch, address generation, and execution stages) |

# One-Word VLESes

**Code**

The following code consists of only one-word VLESes:

```
    align 16
OneWordVlesOnly:
    move.w #01,d1     ; V1, address 1000h
    move.w #02,d2     ; V2, address 1002h
    move.w #03,d3     ; V3, address 1004h
    cmpeq.w #06,d1    ; V4, address 1006h
    bt <D1Select      ; V5, address 1008h
    cmpeq.w #01,d2    ; V6, address 100Ah
    bt <D2Select      ; V7, address 100Ch
    cmpeq.w #02,d3    ; V8, address 100Eh
    bt <D3Select      ; V9, address 1010h
    inc d1            ; V10, address 1012h
    inc d2            ; V11, address 1014h
    inc d3            ; V12, address 1016h
    cmpeq.w #05,d1    ; V13, address 1018h
    bt <D1Select      ; V14, address 101Ah
    cmpeq.w #02,d2    ; V15, address 101Ch
    bt <D2Select      ; V16, address 101Eh
    cmpeq.w #02,d3    ; V17, address 1020h
    bt <D3Select      ; V18, address 1022h
    inc d1            ; V19, address 1024h
    inc d2            ; V20, address 1026h
    inc d3            ; V21, address 1028h
    cmpeq.w #05,d1    ; V22, address 102Ah
    bt <D1Select      ; V23, address 102Ch
    cmpeq.w #02,d2    ; V24, address 102Eh
    bt <D2Select      ; V25, address 1030h
    cmpeq.w #02,d3    ; V26, address 1032h
    bt <D3Select      ; V27, address 1034h
    inc d1            ; V28, address 1036h
    inc d2            ; V29, address 1038h
    inc d3            ; V30, address 103Ah
    cmpeq.w #05,d1    ; V31, address 103Ch
    bt <D1Select      ; V32, address 103Eh
    nop               ; V33, address 1040h
    nop               ; V34, address 1042h
```

**Bus activity**

Notice that once the fetch buffers have been filled, the core accesses program memory only once every eight cycles.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V8 | V7 | V6 | V5 | V4 | V3 | V2 | V1 | | | | |
| 2 | 1020h | V16 | V15 | V14 | V13 | V12 | V11 | V10 | V9 | 1000h | V1 | | |
| 3 | 1030h | V24 | V23 | V22 | V21 | V20 | V19 | V18 | V17 | 1002h | V2 | V1 | |
| 4 | | V32 | V31 | V30 | V29 | V28 | V27 | V26 | V25 | 1004h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 1006h | V4 | V3 | V2 |
| 6 | | – | – | – | – | – | – | – | – | 1008h | V5 | V4 | V3 |
| 7 | | – | – | – | – | – | – | – | – | 100Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 100Ch | V7 | V6 | V5 |
| 9 | | – | – | – | – | – | – | – | – | 100Eh | V8 | V7 | V6 |
| 10 | 1040h | – | – | – | – | – | – | – | – | 1010h | V9 | V8 | V7 |
| 11 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | V34 | V33 | 1012h | V10 | V9 | V8 |
| 12 | | – | – | – | – | – | – | – | – | 1014h | V11 | V10 | V9 |
| 13 | | – | – | – | – | – | – | – | – | 1016h | V12 | V11 | V10 |
| 14 | | – | – | – | – | – | – | – | – | 1018h | V13 | V12 | V11 |
| 15 | | – | – | – | – | – | – | – | – | 101Ah | V14 | V13 | V12 |
| 16 | | – | – | – | – | – | – | – | – | 101Ch | V15 | V14 | V13 |
| 17 | | – | – | – | – | – | – | – | – | 101Eh | V16 | V15 | V14 |
| 18 | 1050h | – | – | – | – | – | – | – | – | 1020h | V17 | V16 | V15 |
| 19 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1022h | V18 | V17 | V16 |
| 20 | | – | – | – | – | – | – | – | – | 1024h | V19 | V18 | V17 |
| 21 | | – | – | – | – | – | – | – | – | 1026h | V20 | V19 | V18 |
| 22 | | – | – | – | – | – | – | – | – | 1028h | V21 | V20 | V19 |
| 23 | | – | – | – | – | – | – | – | – | 102Ah | V22 | V21 | V20 |
| 24 | | – | – | – | – | – | – | – | – | 102Ch | V23 | V22 | V21 |
| 25 | | – | – | – | – | – | – | – | – | 102Eh | V24 | V23 | V22 |
| 26 | 1060h | – | – | – | – | – | – | – | – | 1030h | V25 | V24 | V23 |
| 27 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V26 | V25 | V24 |
| 28 | | – | – | – | – | – | – | – | – | 1034h | V27 | V26 | V25 |
| 29 | | – | – | – | – | – | – | – | – | 1036h | V28 | V27 | V26 |
| 30 | | – | – | – | – | – | – | – | – | 1038h | V29 | V28 | V27 |
| 31 | | – | – | – | – | – | – | – | – | 103Ah | V30 | V29 | V28 |
| 32 | | – | – | – | – | – | – | – | – | 103Ch | V31 | V30 | V29 |
| 33 | | – | – | – | – | – | – | – | – | 103Eh | V32 | V31 | V30 |
| 34 | 1070h | – | – | – | – | – | – | – | – | 1040h | V33 | V32 | V31 |
| 35 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1042h | V34 | V33 | V32 |

## Eight-Word VLESes That Are Fetch-Set Aligned

**Code**

The following code consists of only eight-word VLESes that are fetch-set aligned (the address of each VLES is a multiple of 16):

```
    align 16
AlignedEightWordVlesOnly:
    [
        tfr d0,d4                       ; V1, address 1000h
        sub d1,d1,d1
        sub d2,d2,d2

        move.l #INPUT,r8
    ]
    [
        sub d3,d3,d3                    ; V2, address 1010h
        sub d5,d5,d5
        sub d0,d0,d0

        move.l #OUTPUT,r9
    ]
    [
        tfr d0,d4                       ; V3, address 1020h
        sub d1,d1,d1
        sub d2,d2,d2

        move.l #INPUT,r10
    ]
    [
        mac d0,d0,d8                    ; V4, address 1030h
        mac d1,d1,d9
        mac d2,d2,d10
        mac d3,d3,d11

        move.4f (r8)+,d0:d1:d2:d3
        moves.4f d0:d1:d2:d3,(r9)+
    ]
    [
        sub #1,d4                       ; V5, address 1040h
        inc d5
        tfr d2,d6
        tfr d3,d7

        move.4f (r8)+,d12:d13:d14:d15
        moves.4f d12:d13:d14:d15,(r9)+
    ]
    [
        mac d0,d12,d8                   ; V6, address 1050h
        mac d1,d13,d9
        cmpeq.w #$1fff,d4

        move.4f (r8)+,d0:d1:d2:d3
        moves.4f d0:d1:d2:d3,(r9)+
    ]
```

```
[
    tfrt d0,d5                      ; V7, address 1060h
    mac d6,d14,d10
    mac d7,d15,d11
    mac d10,d11,d7

    move.4f (r8)+,d12:d13:d14:d15
    moves.4f d0:d1:d2:d3,(r9)+
]
[
    tfr d4,d0                       ; V8, address 1070h
    tfr d5,d1
    sub d2,d2,d2
    sub d3,d3,d3

    moves.4f d8:d9:d10:d11,(r10)
    rts
]
```

**Bus activity**

Notice that the core accesses program memory once per cycle.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Cycle** | **PAB** | **PDB** | | | | | | | | **PC** | **VLES** | **VLES** | **VLES** |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V1 | V1 | V1 | V1 | V1 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V2 | V2 | V2 | V2 | V2 | V2 | V2 | V2 | 1000h | V1 | | |
| 3 | 1030h | V3 | V3 | V3 | V3 | V3 | V3 | V3 | V3 | 1010h | V2 | V1 | |
| 4 | 1040h | V4 | V4 | V4 | V4 | V4 | V4 | V4 | V4 | 1020h | V3 | V2 | V1 |
| 5 | 1050h | V5 | V5 | V5 | V5 | V5 | V5 | V5 | V5 | 1030h | V4 | V3 | V2 |
| 6 | 1060h | V6 | V6 | V6 | V6 | V6 | V6 | V6 | V6 | 1040h | V5 | V4 | V3 |
| 7 | 1070h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V6 | V5 | V4 |
| 8 | 1080h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1060h | V7 | V6 | V5 |
| 9 | 1090h | | | | | | | | | 1070h | V8 | V7 | V6 |

# Eight-Word VLESes That Are Fetch-Set Unaligned

**Code**

With the exception of the first two VLESes, the following code consists of eight-word VLESes that are fetch-set unaligned (the address of each VLES isn't a multiple of 16):

```
    align 16
UnalignEightWordVlesOnly:
    nop                                 ; V1, address 1000h
    nop                                 ; V2, address 1002h
    [
        tfr d0,d4                       ; V3, address 1004h
        sub d1,d1,d1
        sub d2,d2,d2

        move.l #INPUT,r8
    ]
    [
        sub d3,d3,d3                    ; V4, address 1014h
        sub d5,d5,d5
        sub d0,d0,d0

        move.l #OUTPUT,r9
    ]
    [
        tfr d0,d4                       ; V5, address 1024h
        sub d1,d1,d1
        sub d2,d2,d2

        move.l #INPUT,r10
    ]
    [
        mac d0,d0,d8                    ; V6, address 1034h
        mac d1,d1,d9
        mac d2,d2,d10
        mac d3,d3,d11

        move.4f (r8)+,d0:d1:d2:d3
        moves.4f d0:d1:d2:d3,(r9)+
    ]
    [
        sub #1,d4                       ; V7, address 1044h
        inc d5
        tfr d2,d6
        tfr d3,d7

        move.4f (r8)+,d12:d13:d14:d15
        moves.4f d12:d13:d14:d15,(r9)+
    ]
    [
        mac d0,d12,d8                   ; V8, address 1054h
        mac d1,d13,d9
        cmpeq.w #$1fff,d4

        move.4f (r8)+,d0:d1:d2:d3
        moves.4f d0:d1:d2:d3,(r9)+
    ]
```

```
  [
      tfrt d0,d5                        ; V9, address 1064h
      mac d6,d14,d10
      mac d7,d15,d11
      mac d10,d11,d7

     move.4f (r8)+,d12:d13:d14:d15
      moves.4f d0:d1:d2:d3,(r9)+
  ]
  [
      tfr d4,d0                         ; V10, address 1074h
      tfr d5,d1
      sub d2,d2,d2
      sub d3,d3,d3

      moves.4f d8:d9:d10:d11,(r10)
      rts
]
```

**Bus activity**

Notice that even though the VLESes are fetch-set unaligned, the core still accesses program memory once per cycle.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Cycle** | **PAB** | **PDB** | | | | | | | | **PC** | **VLES** | **VLES** | **VLES** |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V3 | V3 | V3 | V3 | V3 | V2 | V1 | | | | |
| 2 | 1020h | V4 | V4 | V4 | V4 | V4 | V4 | V3 | V3 | 1000h | V1 | | |
| 3 | 1030h | V5 | V5 | V5 | V5 | V5 | V5 | V4 | V4 | 1002h | V2 | V1 | |
| 4 | 1040h | V6 | V6 | V6 | V6 | V6 | V6 | V5 | V5 | 1004h | V3 | V2 | V1 |
| 5 | 1050h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1014h | V4 | V3 | V2 |
| 6 | 1060h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | V5 | V4 | V3 |
| 7 | 1070h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1034h | V6 | V5 | V4 |
| 8 | 1080h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1044h | V7 | V6 | V5 |
| 9 | 1090h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | V10 | V10 | 1054h | V8 | V7 | V6 |
| 10 | 10A0h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1064h | V9 | V8 | V7 |
| 11 | 10B0h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1074h | V10 | V9 | V8 |

# Chapter 4: Examples—General Change of Flow

## Overview

**Introduction**

This chapter gives examples of nonsequential program flow. For information on how to interpret these examples, see "About the Examples" on page 16.

**What to notice**

The examples in this chapter illustrate the following points:

- All change-of-flow instructions (branches, jumps, returns, and exceptions) require multiple cycles.
- When the destination of a change of flow is a VLES that crosses a fetch-set boundary, the core introduces a one-cycle penalty.
- For all changes of flow, the core must invalidate and refill all four fetch buffers, which results in a minimum of four sequential program-memory accesses.

**In this chapter**

This chapter covers the following topics:

| Topic | Page |
|---|---|
| Change of Flow to a VLES That Is Contained in a Fetch Set | 24 |
| Change of Flow to a VLES That Crosses a Fetch-Set Boundary | 26 |

# Change of Flow to a VLES That Is Contained in a Fetch Set

**Code**

The following code includes changes of flow (V7 and V25) to VLESes (V20 and V8) that don't cross fetch-set boundaries:

```
    align 16
FunctionCallingAligned:
    move.l #INPUT,r0                ; V1, address 1000h
    move.l #INPUT+24,r1             ; V2, address 1006h
    [
        push d6                     ; V3, address 100Ch
        push d7
    ]
    di                              ; V4, address 1012h
    inc d15                         ; V5, address 1014h
    ei                              ; V6, address 101Ah
    bsr >AlignedSubWith128Vles      ; V7, address 101Ch
    [
        pop d6                      ; V8, address 1020h
        pop d7
    ]
    move.l #OUTPUT,r0               ; V9, address 1026h
    nop                             ; V10, address 102Ch
    moves.f d0,(r0)                 ; V11, address 102Eh
    rts                             ; V12, address 1030h

    align 16
AlignedSubWith128Vles:
    [
        sub d8,d8,d8                ; V20, address 1100h
        sub d9,d9,d9
        sub d10,d10,d10
        sub d11,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        mac d0,d4,d8                ; V21, address 1110h
        mac d0,d5,d9
        mac d1,d6,d10
        mac d2,d7,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        mac d0,d4,d8                ; V22, address 1120h
        mac d0,d5,d9
        mac d1,d6,d10
        mac d2,d7,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
```

```
[
    mac d0,d4,d8                ; V23, address 1130h
    mac d0,d5,d9
    mac d1,d6,d10
    mac d2,d7,d11
]
[
    add d8,d9,d9                ; V24, address 113Ch
    add d10,d11,d11
]
[
    add d9,d11,d0               ; V25, address 1144h
    rts
]
```

**Bus activity**

Notice that a change-of-flow instruction requires multiple cycles because the core must invalidate and refill its fetch buffers. In this example, the BSR instruction requires 4 cycles and the RTS instruction requires 3 cycles.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V3 | V2 | V2 | V2 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V7 | V7 | V6 | V5 | V5 | V5 | V4 | V3 | 1000h | V1 | | |
| 3 | 1030h | V11 | V10 | V9 | V9 | V9 | V8 | V8 | V8 | 1006h | V2 | V1 | |
| 4 | | | | | | | | | V12 | 100Ch | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1012h | V4 | V3 | V2 |
| 6 | | | | | | | | | | 1014h | V5 | V4 | V3 |
| 7 | | – | – | – | – | – | – | – | – | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Ch | V7 | V6 | V5 |
| 9 | 1050h | – | – | – | – | – | – | – | – | 1020h | | V7 | V6 |
| 10 | 1100h | | | | | | | | | 1020h | | | V7 |
| 11 | 1110h | V20 | V20 | V20 | V20 | V20 | V20 | V20 | V20 | 1020h | | | |
| 12 | 1120h | V21 | V21 | V21 | V21 | V21 | V21 | V21 | V21 | 1100h | V20 | | |
| 13 | 1130h | V22 | V22 | V22 | V22 | V22 | V22 | V22 | V22 | 1110h | V21 | V20 | |
| 14 | 1140h | V24 | V24 | V23 | V23 | V23 | V23 | V23 | V23 | 1120h | V22 | V21 | V20 |
| 15 | 1150h | | | V25 | V25 | V25 | V25 | V24 | V24 | 1130h | V23 | V22 | V21 |
| 16 | 1160h | | | | | | | | | 113Ch | V24 | V23 | V22 |
| 17 | 1170h | | | | | | | | | 1144h | V25 | V24 | V23 |
| 18 | 1020h | | | | | | | | | 114Ch | | V25 | V24 |
| 19 | 1030h | V11 | V10 | V9 | V9 | V9 | V8 | V8 | V8 | 114Ch | | | V25 |
| 20 | 1040h | | | | | | | | V12 | 1020h | V8 | | |
| 21 | 1050h | | | | | | | | | 1026h | V9 | V8 | |
| 22 | | | | | | | | | | 102Ch | V10 | V9 | V8 |
| 23 | | – | – | – | – | – | – | – | – | 102Eh | V11 | V10 | V9 |
| 24 | 1060h | – | – | – | – | – | – | – | – | 1030h | V12 | V11 | V10 |

# Change of Flow to a VLES That Crosses a Fetch-Set Boundary

**Code**

The following code includes changes of flow (V6 and V25) to VLESes (V20 and V7) that cross fetch-set boundaries:

```
    align 16
FunctionCallingUnalign:
    move.l #INPUT,r0                ; V1, address 1000h
    move.l #INPUT+24,r1             ; V2, address 1006h
    [
        push d6                     ; V3, address 100Ch
        push d7
    ]
    di                              ; V4, address 1012h
    inc d15                         ; V5, address 1014h
    bsr >UnalignSubWith128Vles      ; V6, address 101Ah
    [
        pop d6                      ; V7, address 101Eh
        pop d7
    ]
    move.l #OUTPUT,r0               ; V8, address 1024h
    ei                              ; V9, address 102Ah
    moves.f d0,(r0)                 ; V10, address 102Ch
    rts                             ; V11, address 102Eh

    align 16
UnalignSubWith128Vles:
    [
        sub d8,d8,d8                ; V20, address 1108h
        sub d9,d9,d9
        sub d10,d10,d10
        sub d11,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        mac d0,d4,d8                ; V21, address 1118h
        mac d0,d5,d9
        mac d1,d6,d10
        mac d2,d7,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        mac d0,d4,d8                ; V22, address 1128h
        mac d0,d5,d9
        mac d1,d6,d10
        mac d2,d7,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
```

```
[
    mac d0,d4,d8                ; V23, address 1138h
    mac d0,d5,d9
    mac d1,d6,d10
    mac d2,d7,d11
]
[
    add d8,d9,d9                ; V24, address 1144h
    add d10,d11,d11
]
[
    add d9,d11,d0               ; V25, address 114Ch
    rts
]
```

**Bus activity**

Notice that a change of flow to a VLES that crosses a fetch-set boundary introduces an additional one-cycle penalty.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V3 | V2 | V2 | V2 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V3 | 1000h | V1 | | |
| 3 | 1030h | V11 | V10 | V9 | V8 | V8 | V8 | V7 | V7 | 1006h | V2 | V1 | |
| 4 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | 100Ch | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1012h | V4 | V3 | V2 |
| 6 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1014h | V5 | V4 | V3 |
| 7 | | – | – | – | – | – | – | – | – | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Eh | | V6 | V5 |
| 9 | 1100h | – | – | – | – | – | – | – | – | 101Eh | | | V6 |
| 10 | 1110h | V20 | V20 | V20 | V20 | ▨ | ▨ | ▨ | ▨ | 101Eh | | | |
| 11 | 1120h | V21 | V21 | V21 | V21 | V20 | V20 | V20 | V20 | 101Eh | | | |
| 12 | 1130h | V22 | V22 | V22 | V22 | V21 | V21 | V21 | V21 | 1108h | V20 | | |
| 13 | 1140h | V23 | V23 | V23 | V23 | V22 | V22 | V22 | V22 | 1118h | V21 | V20 | |
| 14 | 1150h | V25 | V25 | V24 | V24 | V24 | V24 | V23 | V23 | 1128h | V22 | V21 | V20 |
| 15 | 1160h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | V25 | V25 | 1138h | V23 | V22 | V21 |
| 16 | 1170h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1144h | V24 | V23 | V22 |
| 17 | 1170h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 114Ch | V25 | V24 | V23 |
| 18 | 1010h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1154h | | V25 | V24 |
| 19 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V3 | 1154h | | | V25 |
| 20 | 1030h | V11 | V10 | V9 | V8 | V8 | V8 | V7 | V7 | 101Eh | | | |
| 21 | 1040h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 101Eh | V7 | | |
| 22 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1024h | V8 | V7 | |
| 23 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 102Ah | V9 | V8 | V7 |
| 24 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 102Ch | V10 | V9 | V8 |
| 25 | | – | – | – | – | – | – | – | – | 102Eh | V11 | V11 | V9 |

# Chapter 5: Examples—Long Loops

## Overview

**Introduction**

This chapter gives examples of long loops. For information on how to interpret these examples, see “About the Examples” on page 16.

**Long loops**

A *long loop* is a hardware-controlled loop that consists of three or more VLESes.

**What to notice**

The examples in this chapter illustrate the following points:

- Long loops require program-memory bus activity.
- For long loops, the core doesn’t introduce any minimum timing penalty, like it does for other changes of flow.
- For long loops in which the first VLES crosses a fetch-set boundary, the core introduces a one-cycle penalty. This is similar to the one-cycle penalty imposed when the destination of a change of flow is a VLES that crosses a fetch-set boundary (see “Change of Flow to a VLES That Crosses a Fetch-Set Boundary” on page 26).
- Because of the change of flow that occurs when one iteration of a long loop ends and another iteration begins, the core must invalidate and refill all four fetch buffers, which results in a minimum of four sequential program-memory accesses.
- For long loops that occupy three or fewer fetch sets, the core fetches some sets that it never uses.

**In this chapter**

This chapter covers the following topics:

| Topic | Page |
|---|---|
| Eight-Word Long Loop—First VLES Is Contained in a Fetch Set | 30 |
| Eight-Word Long Loop—First VLES Crosses a Fetch-Set Boundary | 32 |
| Thirty-two–Word Long Loop—First VLES Is Contained in a Fetch Set | 34 |
| Thirty-two–Word Long Loop—First VLES Crosses a Fetch-Set Boundary | 37 |

## Eight-Word Long Loop—First VLES Is Contained in a Fetch Set

**Code**

The following code includes a four-VLES (V7–V10), eight-word long loop in which the first VLES (V7) is contained in a fetch set:

```
    align 16
FetchContainedFourVlesLoop_128bit:
    [
        dosetup0 LoopA                ; V1, address 1000h
        doen0 #7
    ]
    move.l #INPUT,r0                  ; V2, address 1008h
    move.l #INPUT+4,r1                ; V3, address 100Eh
    move.l #OUTPUT,r7                 ; V4, address 1014h
    move.l #OUTPUT+4,r3               ; V5, address 101Ah
    [
        move.4f (r0)+,d0:d1:d2:d3     ; V6, address 1020h
        move.4f (r1)+,d4:d5:d6:d7
    ]
LoopA:
    loopstart0
    [
        add d0,d4,d8                  ; V7, address 1024h
        moves.4f d0:d1:d2:d3,(r7)+
    ]
    moves.4f d4:d5:d6:d7,(r3)+        ; V8, address 102Ch
    move.4f (r0)+,d0:d1:d2:d3         ; V9, address 1030h
    move.4f (r1)+,d4:d5:d6:d7         ; V10, address 1032h
    loopend0
    [
        add d0,d4,d8                  ; V11, address 1034h
        add d1,d5,d9
        add d2,d6,d10
        add d3,d7,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
   rts                                ; V12, address 1044h
```

**Bus activity**

Notice that the core doesn't introduce any timing penalties because the first VLES in the loop doesn't cross a fetch-set boundary. Also notice that, because the loop occupies three or fewer fetch sets, the core fetches some sets that it never uses.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V2 | V2 | V2 | V1 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V5 | V5 | V5 | V4 | V4 | V4 | V3 | V3 | 1000h | V1 | | |
| 3 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1008h | V2 | V1 | |
| 4 | | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 100Eh | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1014h | V4 | V3 | V2 |
| 6 | | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 101Ah | V5 | V4 | V3 |
| 7 | 1050h | – | – | – | – | – | – | – | – | 1020h | V6 | V5 | V4 |
| 8 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1024h | V7 | V6 | V5 |
| 9 | | – | – | – | – | – | – | – | – | 102Ch | V8 | V7 | V6 |
| 10 | 1020h | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V7 |
| 11 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 12 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 13 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 14 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 15 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 16 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 17 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 18 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 19 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 20 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 21 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 22 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 23 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 24 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 25 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 26 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 27 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 28 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 29 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 30 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 31 | 1030h | V8 | V8 | V7 | V7 | V7 | V7 | V6 | V6 | 1032h | V10 | V9 | V8 |
| 32 | 1040h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V9 | 1024h | V7 | V10 | V9 |
| 33 | 1050h | ▨ | ▨ | ▨ | ▨ | ▨ | V12 | V11 | V11 | 102Ch | V8 | V7 | V10 |
| 34 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V9 | V8 | V7 |
| 35 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V10 | V9 | V8 |
| 36 | | – | – | – | – | – | – | – | – | 1034h | V11 | V10 | V9 |
| 37 | | – | – | – | – | – | – | – | – | 1044h | V12 | V11 | V10 |

# Eight-Word Long Loop—First VLES Crosses a Fetch-Set Boundary

**Code**

The following code includes a four-VLES (V10–V13), eight-word long loop in which the first VLES (V10) crosses a fetch-set boundary:

```
    align 16
FetchCrossingFourVlesLoop_128bit:
    [
        dosetup0 LoopA               ; V1, address 1000h
        doen0 #7
    ]
    move.l #INPUT,r0                 ; V2, address 1008h
    move.l #INPUT+4,r1               ; V3, address 100Eh
    move.l #OUTPUT,r7                ; V4, address 1014h
    move.l #OUTPUT+4,r3              ; V5, address 101Ah
    [
        move.4f (r0)+,d0:d1:d2:d3    ; V6, address 1020h
        move.4f (r1)+,d4:d5:d6:d7
    ]
    nop                              ; V7, address 1024h
    nop                              ; V8, address 1026h
    nop                              ; V9, address 1028h
LoopA:
    loopstart0
    [
        add d0,d4,d8                 ; V10, address 102A
        moves.4f d0:d1:d2:d3,(r7)+
    ]
    moves.4f d4:d5:d6:d7,(r3)+       ; V11, address 1032h
    move.4f (r0)+,d0:d1:d2:d3        ; V12, address 1036h
    move.4f (r1)+,d4:d5:d6:d7        ; V13, address 1038h
    loopend0
    [
        add d0,d4,d8                 ; V14, address 103Ah
        add d1,d5,d9
        add d2,d6,d10
        add d3,d7,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    rts                              ; V15, address 104Ah
```

**Bus activity**

Notice that the core introduces a one-cycle penalty because the first VLES in the loop crosses a fetch-set boundary. Also notice that, because the loop occupies three or fewer fetch sets, the core fetches some sets that it never uses.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V2 | V2 | V2 | V1 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V5 | V5 | V5 | V4 | V4 | V4 | V3 | V3 | 1000h | V1 | | |
| 3 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1008h | V2 | V1 | |
| 4 | | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 100Eh | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1014h | V4 | V3 | V2 |
| 6 | | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 101Ah | V5 | V4 | V3 |
| 7 | 1050h | – | – | – | – | – | – | – | – | 1020h | V6 | V5 | V4 |
| 8 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1024h | V7 | V6 | V5 |
| 9 | | – | – | – | – | – | – | – | – | 1026h | V8 | V7 | V6 |
| 10 | | – | – | – | – | – | – | – | – | 1028h | V9 | V8 | V7 |
| 11 | | – | – | – | – | – | – | – | – | 102Ah | V10 | V9 | V8 |
| 12 | 1060h | – | – | – | – | – | – | – | – | 1032h | V11 | V10 | V9 |
| 13 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 14 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 15 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 16 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 17 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 18 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 19 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 20 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 21 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 22 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 23 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 24 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 25 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 26 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 27 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 28 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 29 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 30 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 31 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 32 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 33 | 1020h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 34 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 35 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 36 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 37 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 38 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 39 | 1030h | V10 | V10 | V10 | V9 | V8 | V7 | V6 | V6 | 1038h | V13 | V12 | V11 |
| 40 | 1040h | V14 | V14 | V14 | V13 | V12 | V11 | V11 | V10 | 102Ah | | V13 | V12 |
| 41 | 1050h | ▨ | ▨ | V15 | V14 | V14 | V14 | V14 | V14 | 102Ah | V10 | | V13 |
| 42 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1032h | V11 | V10 | |
| 43 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1036h | V12 | V11 | V10 |
| 44 | | – | – | – | – | – | – | – | – | 1038h | V13 | V12 | V11 |
| 45 | | – | – | – | – | – | – | – | – | 1039h | V14 | V13 | V12 |
| 46 | 1070h | – | – | – | – | – | – | – | – | 1040h | V15 | V14 | V13 |

## Thirty-two–Word Long Loop—First VLES Is Contained in a Fetch Set

**Code**

The following code includes a long loop that consists of four eight-word VLESes (V7–V10). In this loop, the first VLES (V7) is contained in a fetch set.

```
    align 16
AlignedFourVlesLoop_512bit:
    [
        dosetup0 LoopC                 ; V1, address 1000h
        doen0 #7
    ]
        move.l #INPUT,r0               ; V2, address 1008h
        move.l #OUTPUT,r7              ; V3, address 100Eh
        adda #4,r0,r1                  ; V4, address 1014h
        adda #4,r7,r3                  ; V5, address 1018h
    [
        move.4f (r0)+,d0:d1:d2:d3      ; V6, address 101Ch
        move.4f (r1)+,d4:d5:d6:d7
    ]
LoopC:
    loopstart0
    [
        add d0,d8,d8                   ; V7, address 1020h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                   ; V8, address 1030h
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        add d0,d8,d8                   ; V9, address 1040h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
```

```
[
    add d4,d8,d8                  ; V10, address 1050h
    add d5,d9,d9
    add d6,d10,d10
    add d7,d11,d11

    move.4f (r0)+,d0:d1:d2:d3
    move.4f (r1)+,d4:d5:d6:d7
]
loopend0
[
    add d0,d8,d8                  ; V11, address 1060h
    add d1,d9,d9
    add d2,d10,d10
    add d3,d11,d11

    moves.4f d0:d1:d2:d3,(r7)+
    moves.4f d4:d5:d6:d7,(r3)+
]
[
    add d4,d8,d8                  ; V12, address 1070h
    add d5,d9,d9
    add d6,d10,d10
    add d7,d11,d11
]
rts                               ; V13, address 107Ch
```

**Bus activity**

Notice that the core doesn't introduce any timing penalties because the first VLES in the loop doesn't cross a fetch-set boundary. Also notice that, because the loop occupies more than three fetch sets, the core uses all of the sets that it fetches.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V2 | V2 | V2 | V1 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V6 | V6 | V5 | V5 | V4 | V4 | V3 | V3 | 1000h | V1 | | |
| 3 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1008h | V2 | V1 | |
| 4 | | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 100Eh | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1014h | V4 | V3 | V2 |
| 6 | | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1018h | V5 | V4 | V3 |
| 7 | | – | – | – | – | – | – | – | – | 101Ch | V6 | V5 | V4 |
| 8 | 1050h | – | – | – | – | – | – | – | – | 1020h | V7 | V6 | V5 |
| 9 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1030h | V8 | V7 | V6 |
| 10 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V11 | V11 | 1040h | V9 | V8 | V7 |
| 11 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 12 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 13 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 14 | 1020h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 15 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 16 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 17 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 18 | 1020h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 19 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 20 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 21 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 22 | 1020h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 23 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 24 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 25 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 26 | 1020h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 27 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 28 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 29 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 30 | 1020h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 31 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1050h | V10 | V9 | V8 |
| 32 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1020h | V7 | V10 | V9 |
| 33 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1030h | V8 | V7 | V10 |
| 34 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 1040h | V9 | V8 | V7 |
| 35 | 1070h | V11 | V11 | V11 | V11 | V11 | V11 | V11 | V11 | 1050h | V10 | V9 | V8 |
| 36 | 1080h | | V13 | V12 | V12 | V12 | V12 | V12 | V12 | 1060h | V11 | V10 | V9 |
| 37 | 1090h | | | | | | | | | 1070h | V12 | V11 | V10 |
| 38 | 10A0h | | | | | | | | | 107Ch | V13 | V12 | V11 |

# Thirty-two–Word Long Loop—First VLES Crosses a Fetch-Set Boundary

**Code**

The following code includes a long loop that consists of four eight-word VLESes (V7–V10). In this loop, the first VLES (V7) crosses a fetch-set boundary.

```
    align 16
UnalignFourVlesLoop_512bit:
    [
        dosetup0 LoopC               ; V1, address 1000h
        doen0 #7
    ]
    move.l #INPUT,r0                 ; V2, address 1008h
    move.l #INPUT+4,r1               ; V3, address 100Eh
    move.l #OUTPUT,r7                ; V4, address 1014h
    move.l #OUTPUT+4,r3              ; V5, address 101Ah
    [
        move.4f (r0)+,d0:d1:d2:d3    ; V6, address 1020h
        move.4f (r1)+,d4:d5:d6:d7
    ]
LoopC:
    loopstart0
    [
        add d0,d8,d8                 ; V7, address 1024h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                 ; V8, address 1034h
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    [
        add d0,d8,d8                 ; V9, address 1044h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
```

```
[
    add d4,d8,d8                    ; V10, address 1054h
    add d5,d9,d9
    add d6,d10,d10
    add d7,d11,d11

    move.4f (r0)+,d0:d1:d2:d3
    move.4f (r1)+,d4:d5:d6:d7
]
loopend0
[
    add d0,d8,d8                    ; V11, address 1064h
    add d1,d9,d9
    add d2,d10,d10
    add d3,d11,d11

    moves.4f d0:d1:d2:d3,(r7)+
    moves.4f d4:d5:d6:d7,(r3)+
]
[
    add d4,d8,d8                    ; V12, address 1074h
    add d5,d9,d9
    add d6,d10,d10
    add d7,d11,d11
]
rts                                 ; V13, address 1080h
```

**Bus activity**

Notice that the core introduces a one-cycle penalty because the first VLES in the loop crosses a fetch-set boundary. Also notice that, because the loop occupies more than three fetch sets, the core uses all of the sets that it fetches.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V3 | V2 | V2 | V2 | V1 | V1 | V1 | V1 | | | | |
| 2 | 1020h | V5 | V5 | V5 | V4 | V4 | V4 | V3 | V3 | 1000h | V1 | | |
| 3 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1008h | V2 | V1 | |
| 4 | | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 100Eh | V3 | V2 | V1 |
| 5 | 1040h | – | – | – | – | – | – | – | – | 1014h | V4 | V3 | V2 |
| 6 | | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 101Ah | V5 | V4 | V3 |
| 7 | | – | – | – | – | – | – | – | – | 1020h | V6 | V5 | V4 |
| 8 | 1050h | – | – | – | – | – | – | – | – | 1024h | V7 | V6 | V5 |
| 9 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | V6 |
| 10 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 11 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 12 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 13 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 14 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 15 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 16 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 17 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 18 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 19 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 20 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 21 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 22 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 23 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 24 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 25 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 26 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 27 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 28 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 29 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 30 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 31 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 32 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 33 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 34 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 35 | 1020h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 36 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V6 | V6 | 1055h | V10 | V9 | V8 |
| 37 | 1040h | V8 | V8 | V8 | V8 | V8 | V8 | V7 | V7 | 1024h | | V10 | V9 |
| 38 | 1050h | V9 | V9 | V9 | V9 | V9 | V9 | V8 | V8 | 1024h | V7 | | V10 |
| 39 | 1060h | V10 | V10 | V10 | V10 | V10 | V10 | V9 | V9 | 1034h | V8 | V7 | |
| 40 | 1070h | V11 | V11 | V11 | V11 | V11 | V11 | V10 | V10 | 1044h | V9 | V8 | V7 |
| 41 | 1080h | V12 | V12 | V12 | V12 | V12 | V12 | V11 | V11 | 1055h | V10 | V9 | V8 |
| 42 | 1090h | | | | | | | | V13 | 1064h | V11 | V10 | V9 |
| 43 | 10A0h | | | | | | | | | 1074h | V12 | V11 | V10 |
| 44 | 10B0h | | | | | | | | | 1080h | V13 | V12 | V11 |

# Chapter 6: Examples—Short Loops

## Overview

**Introduction**

This chapter gives examples of short loops. For information on how to interpret these examples, see “About the Examples” on page 16.

**Short loops**

A *short loop* is a hardware-controlled loop that consists of one or two VLESes.

**What to notice**

The examples in this chapter illustrate the following points:

- Short loops require no program-memory bus activity after the first iteration. The core reuses the instructions in the fetch buffers instead of repeatedly fetching them from program memory. Therefore, short loops reduce power consumption and further decrease the likelihood of memory-access conflicts.
- For short loops, the core doesn’t introduce any timing penalties, regardless of whether the first VLES in the loop crosses a fetch-set boundary.

**In this chapter**

This chapter covers the following topics:

## Eight-Word Short Loop—First VLES Is Contained in a Fetch Set

**Code**

The following code includes a one-VLES (V5), eight-word short loop in which the first (and only) VLES (V5) is contained in a fetch set:

```
    align 16
FetchContainedOneVlesLoop_128bit:
    doensh0 #7                       ; V1, address 1000h
    move.l #INPUT,r0                 ; V2, address 1006h
    move.l #OUTPUT,r7                ; V3, address 100Ch
    move.4w (r0)+,d0:d1:d2:d3        ; V4, address 100Eh
    loopstart0
    [
        add d0,d8,d8                 ; V5, address 1010h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        move.4f (r0)+,d0:d1:d2:d3
    ]
    loopend0
    [
        add d0,d8,d8                 ; V6, address 1020h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
    ]
    [
        add d8,d9,d9                 ; V7, address 102Eh
        add d10,d11,d11
    ]
    add d10,d11,d11                  ; V8, address 1036h
    rts                              ; V9, address 103Ch
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn’t introduce any timing penalties.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Cycle** | **PAB** | **PDB** | | | | | | | | **PC** | **VLES** | **VLES** | **VLES** |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V5 | V5 | V5 | V5 | V5 | V5 | V5 | V5 | 1000h | V1 | | |
| 3 | 1030h | V7 | V6 | V6 | V6 | V6 | V6 | V6 | V6 | 1002h | V2 | V1 | |
| 4 | | ▨ | V9 | V8 | V8 | V8 | V7 | V7 | V7 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1010h | V5 | V4 | V3 |
| 7 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1010h | V5 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 1010h | V5 | V5 | V5 |
| 9 | | – | – | – | – | – | – | – | – | 1010h | V5 | V5 | V5 |
| 10 | | – | – | – | – | – | – | – | – | 1010h | V5 | V5 | V5 |
| 11 | | – | – | – | – | – | – | – | – | 1010h | V5 | V5 | V5 |
| 12 | | – | – | – | – | – | – | – | – | 1010h | V5 | V5 | V5 |
| 13 | 1050h | – | – | – | – | – | – | – | – | 1020h | V6 | V5 | V5 |
| 14 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 102Eh | V7 | V6 | V5 |
| 15 | 1060h | – | – | – | – | – | – | – | – | 1036h | V8 | V7 | V6 |
| 16 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 103Ch | V9 | V8 | V7 |

# Eight-Word Short Loop—First VLES Crosses a Fetch Set Boundary

**Code**

The following code includes a one-VLES (V6), eight-word short loop in which the first (and only) VLES (V6) crosses a fetch-set boundary:

```
    align 16
FetchCrossingOneVlesLoop_128bit:
    doensh0 #7                      ; V1, address 1000h
    move.l #INPUT,r0                ; V2, address 1002h
    move.l #OUTPUT,r7               ; V3, address 1008h
    move.4w (r0)+,d0:d1:d2:d3       ; V4, address 100Eh
    nop                             ; V5, address 1010h
    loopstart0
    [
        add d0,d8,d8                ; V6, address 1012h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        move.4f (r0)+,d0:d1:d2:d3
    ]
    loopend0
    [
        add d0,d8,d8                ; V7, address 1022h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
    ]
    [
        add d8,d9,d9                ; V8, address 1030h
        add d10,d11,d11
    ]
    add d10,d11,d11                 ; V9, address 1038h
    rts                             ; V10, address 103Eh
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn’t introduce any timing penalties (as it would for a long loop in which the first VLES crosses a fetch-set boundary).

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V6 | V6 | V6 | V6 | V6 | V6 | V6 | V5 | 1000h | V1 | | |
| 3 | 1030h | V7 | V7 | V7 | V7 | V7 | V7 | V7 | V6 | 1002h | V2 | V1 | |
| 4 | | V10 | V9 | V9 | V9 | V8 | V8 | V8 | V8 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1010h | V5 | V4 | V3 |
| 7 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1012h | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V5 |
| 9 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V6 |
| 10 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V6 |
| 11 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V6 |
| 12 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V6 |
| 13 | | – | – | – | – | – | – | – | – | 1012h | V6 | V6 | V6 |
| 14 | 1050h | – | – | – | – | – | – | – | – | 1022h | V7 | V6 | V6 |
| 15 | 1060h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1030h | V8 | V7 | V6 |
| 16 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1038h | V9 | V8 | V7 |
| 17 | | – | – | – | – | – | – | – | – | 103Eh | V10 | V9 | V8 |

## Five-Word Short Loop—First VLES Is Contained in a Fetch Set

**Code**

The following code includes a two-VLES (V9 and V10), five-word short loop in which the first VLES (V9) is contained in a fetch set:

```
    align 16
FetchContainedTwoVlesLoop_80bit:
    doensh0 #7                       ; V1, address 1000h
    move.l #INPUT,r0                 ; V2, address 1002h
    move.l #INPUT+4,r1               ; V3, address 1008h
    move.l #OUTPUT,r7                ; V4, address 100Eh
    move.l #OUTPUT+4,r3              ; V5, address 1014h
    [
        move.4f (r0)+,d0:d1:d2:d3    ; V6, address 101Ah
        move.4f (r1)+,d4:d5:d6:d7
    ]
    nop                              ; V7, address 101Eh
    nop                              ; V8, address 1020h
    loopstart0
    [
        moves.4f d0:d1:d2:d3,(r7)+   ; V9, address 1022h
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        move.4f (r0)+,d0:d1:d2:d3    ; V10, address 1028h
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopend0
    [
        add d0,d4,d8                 ; V11, address 102Ch
        add d1,d5,d9
        add d2,d6,d10
        add d3,d7,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    rts                              ; V12, address 103Ch
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn't introduce any timing penalties.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V4 | 1000h | V1 | | |
| 3 | 1030h | V11 | V11 | V10 | V10 | V9 | V9 | V9 | V8 | 1002h | V2 | V1 | |
| 4 | | ▨ | V12 | V11 | V11 | V11 | V11 | V11 | V11 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1014h | V5 | V4 | V3 |
| 7 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V6 | V5 |
| 9 | 1050h | – | – | – | – | – | – | – | – | 1020h | V8 | V7 | V6 |
| 10 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1022h | V9 | V8 | V7 |
| 11 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V8 |
| 12 | | – | – | – | – | – | – | – | – | 1022h | V9 | V10 | V9 |
| 13 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 14 | | – | – | – | – | – | – | – | – | 1022h | V9 | V10 | V9 |
| 15 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 16 | | – | – | – | – | – | – | – | – | 1038h | V9 | V10 | V9 |
| 17 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 18 | | – | – | – | – | – | – | – | – | 1038h | V9 | V10 | V9 |
| 19 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 20 | | – | – | – | – | – | – | – | – | 1038h | V9 | V10 | V9 |
| 21 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 22 | | – | – | – | – | – | – | – | – | 1038h | V9 | V10 | V9 |
| 23 | | – | – | – | – | – | – | – | – | 1028h | V10 | V9 | V10 |
| 24 | | – | – | – | – | – | – | – | – | 102Ch | V11 | V10 | V9 |
| 25 | 1060h | – | – | – | – | – | – | – | – | 103Ch | V12 | V11 | V10 |

# Five-Word Short Loop—First VLES Crosses a Fetch-Set Boundary

**Code**

The following code includes a two-VLES (V7 and V8), five-word short loop in which the first VLES (V7) crosses a fetch-set boundary:

```
    align 16
FetchCrossingTwoVlesLoop_80bit:
    doensh0 #7                         ; V1, address 1000h
    move.l #INPUT,r0                   ; V2, address 1002h
    move.l #INPUT+4,r1                 ; V3, address 1008h
    move.l #OUTPUT,r7                  ; V4, address 100Eh
    move.l #OUTPUT+4,r3                ; V5, address 1014h
    [
        move.4f (r0)+,d0:d1:d2:d3      ; V6, address 101Ah
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopstart0
    [
        moves.4f d0:d1:d2:d3,(r7)+     ; V7, address 101Eh
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        move.4f (r0)+,d0:d1:d2:d3      ; V8, address 1024h
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopend0
    [
        add d0,d4,d8                   ; V9, address 1028h
        add d1,d5,d9
        add d2,d6,d10
        add d3,d7,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    rts                                ; V10, address 1038h
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn't introduce any timing penalties (as it would for a long loop in which the first VLES crosses a fetch-set boundary).

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V4 | 1000h | V1 | | |
| 3 | 1030h | V9 | V9 | V9 | V9 | V8 | V8 | V7 | V7 | 1002h | V2 | V1 | |
| 4 | | ▨ | ▨ | ▨ | V10 | V9 | V9 | V9 | V9 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1014h | V5 | V4 | V3 |
| 7 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V6 | V5 |
| 9 | 1050h | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V6 |
| 10 | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 101Eh | V7 | V8 | V7 |
| 11 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 12 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 13 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 14 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 15 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 16 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 17 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 18 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 19 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 20 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 21 | | – | – | – | – | – | – | – | – | 1024h | V8 | V7 | V8 |
| 22 | | – | – | – | – | – | – | – | – | 1028h | V9 | V8 | V7 |
| 23 | 1060h | – | – | – | – | – | – | – | – | 1038h | V10 | V9 | V8 |

# Sixteen-Word Short Loop—First VLES Is Contained in a Fetch Set

**Code**

The following code includes a short loop that consists of two eight-word VLESes (V8 and V9). In this loop, the first VLES (V8) is contained in a fetch set.

```
    align 16
FetchContainedTwoVlesLoop_256bit:
    doensh0 #7                        ; V1, address 1000h
    move.l #INPUT,r0                  ; V2, address 1002h
    move.l #INPUT+4,r1                ; V3, address 1008h
    move.l #OUTPUT,r7                 ; V4, address 100Eh
    move.l #OUTPUT+4,r3               ; V5, address 1014h
    [
        move.4f (r0)+,d0:d1:d2:d3     ; V6, address 101Ah
        move.4f (r1)+,d4:d5:d6:d7
    ]
    nop                               ; V7, address 101Eh
    loopstart0
    [
        add d0,d8,d8                  ; V8, address 1020h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                  ; V9, address 1030h
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopend0
    [
        add d0,d8,d8                  ; V10, address 1040h
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                  ; V11, address 1050h
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11
    ]
    rts                               ; V12, address 105Ch
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn't introduce any timing penalties.

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V4 | 1000h | V1 | | |
| 3 | 1030h | V8 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1002h | V2 | V1 | |
| 4 | | V9 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1014h | V5 | V4 | V3 |
| 7 | | V10 | V10 | V10 | V10 | V10 | V10 | V10 | V10 | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V6 | V5 |
| 9 | 1050h | – | – | – | – | – | – | – | – | 1020h | V8 | V7 | V6 |
| 10 | 1060h | | V12 | V11 | V11 | V11 | V11 | V11 | V11 | 1030h | V9 | V8 | V7 |
| 11 | | | | | | | | | | 1020h | V8 | V9 | V8 |
| 12 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 13 | | – | – | – | – | – | – | – | – | 1020h | V8 | V9 | V8 |
| 14 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 15 | | – | – | – | – | – | – | – | – | 1020h | V8 | V9 | V8 |
| 16 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 17 | | – | – | – | – | – | – | – | – | 1020h | V8 | V9 | V8 |
| 18 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 19 | | – | – | – | – | – | – | – | – | 1020h | V8 | V9 | V8 |
| 20 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 21 | | – | – | – | – | – | – | – | – | 1020h | V8 | V9 | V8 |
| 22 | | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V9 |
| 23 | 1070h | – | – | – | – | – | – | – | – | 1038h | V10 | V9 | V8 |
| 24 | 1080h | | | | | | | | | 1050h | V11 | V10 | V9 |
| 25 | | | | | | | | | | 105Ch | V12 | V11 | V10 |

## Sixteen-Word Short Loop—First VLES Crosses a Fetch-Set Boundary

**Code**

The following code includes a short loop that consists of two eight-word VLESes (V7 and V8). In this loop, the first VLES (V7) crosses a fetch-set boundary.

```
    align 16
FetchCrossingTwoVlesLoop_256bit:
    doensh0 #7                        ; V1, address 1000h
    move.l #INPUT,r0                  ; V2, address 1002h
    move.l #INPUT+4,r1                ; V3, address 1008h
    move.l #OUTPUT,r7                 ; V4, address 100Eh
    move.l #OUTPUT+4,r3               ; V5, address 1014h
    [
        move.4f (r0)+,d0:d1:d2:d3     ; V6, address 101Ah
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopstart0
    [
        add d0,d8,d8                  ; V7, address 101Eh
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                  ; V8, address 102Eh
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11

        move.4f (r0)+,d0:d1:d2:d3
        move.4f (r1)+,d4:d5:d6:d7
    ]
    loopend0
    [
        add d0,d8,d8                  ; V9, address 103Eh
        add d1,d9,d9
        add d2,d10,d10
        add d3,d11,d11

        moves.4f d0:d1:d2:d3,(r7)+
        moves.4f d4:d5:d6:d7,(r3)+
    ]
    [
        add d4,d8,d8                  ; V10, address 104Eh
        add d5,d9,d9
        add d6,d10,d10
        add d7,d11,d11
    ]
    rts                               ; V11, address 104Ah
```

**Bus activity**

Notice that there is no program-memory bus activity after the first iteration of the loop. Also notice that the core doesn't introduce any timing penalties (as it would for a long loop in which the first VLES crosses a fetch-set boundary).

| | 1 | 2 | | | | | | | | 3 | | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle | PAB | PDB | | | | | | | | PC | VLES | VLES | VLES |
| 0 | 1000h | – | – | – | – | – | – | – | – | | | | |
| 1 | 1010h | V4 | V3 | V3 | V3 | V2 | V2 | V2 | V1 | | | | |
| 2 | 1020h | V7 | V6 | V6 | V5 | V5 | V5 | V4 | V4 | 1000h | V1 | | |
| 3 | 1030h | V8 | V7 | V7 | V7 | V7 | V7 | V7 | V7 | 1002h | V2 | V1 | |
| 4 | | V9 | V8 | V8 | V8 | V8 | V8 | V8 | V8 | 1008h | V3 | V2 | V1 |
| 5 | | – | – | – | – | – | – | – | – | 100Eh | V4 | V3 | V2 |
| 6 | 1040h | – | – | – | – | – | – | – | – | 1014h | V5 | V4 | V3 |
| 7 | | V10 | V9 | V9 | V9 | V9 | V9 | V9 | V9 | 101Ah | V6 | V5 | V4 |
| 8 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V6 | V5 |
| 9 | 1050h | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V6 |
| 10 | | ▨ | ▨ | V11 | V12 | V12 | V12 | V12 | V12 | 101Eh | V7 | V8 | V7 |
| 11 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 12 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 13 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 14 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 15 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 16 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 17 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 18 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 19 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 20 | | – | – | – | – | – | – | – | – | 101Eh | V7 | V8 | V7 |
| 21 | | – | – | – | – | – | – | – | – | 102Eh | V8 | V7 | V8 |
| 22 | 1060h | – | – | – | – | – | – | – | – | 1030h | V9 | V8 | V7 |
| 23 | 1070h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1038h | V10 | V9 | V8 |
| 24 | 1080h | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | 1050h | V11 | V10 | V9 |