Documentation Développeurs
Apple Computer France 1987

Document développeur numéro 1

Description of 65816 flow of control instructions, Jump Absolute, Block move

type d'upgrade de ce ducument : 1
1 Documentation de première catégorie inchangée
2 Documentation de deuxième catégorie mise à jour
3 Documentation de deuxième catégorie inchangée
4 Mise à jour payante de la documentation de première catégorie
5 Mise à jour gratuite de la documentation de première catégorie
6 Nouveautés payantes non vitales
7 Nouveautés gratuites et vitales

Taille : 5 page(s) environ

Domaine : 816

VERSION :
DATE : 5.06.85

James Jatczynski Consulting □ MEMO

Description of 65816 Flow of Control Instructions

To: Columbia Software

From: Jim Jatczynski 

Date: 28 May 1985

The attached document describes the 65816 flow of control instructions including branches, jumps, subroutine jumps and returns, and software interrupts and returns. It fills in details missing from the WDC and GTE data sheets and corrects errors therein. This document is intended to be the most correct and up to date description of these instructions. In cases where this document contradicts the data sheets, the data sheets are wrong.

If you see any errors or think additional information should be included, please let me know as soon as possible

James Jatczynski Consulting □ MEMO

Clarification of 65816 Jump Absolute and Subroutine Jump Absolute

To: Columbia Software

From: Jim Jatczynski 

Date: 21 May 1985

The GTE Advance Information data sheet on the G65SC816 defines the absolute addressing mode as follows:

> With Absolute addressing the second and third bytes of the instruction form the low-order 16 bits of the effective address. The Data Bank Register contains the high-order 8 bits of the operand address.

This description is incorrect for the Jump Absolute (opcode $4C) and the Subroutine Jump Absolute (opcode $20) instructions. For these instructions, the *Program Bank Register* contains the high-order 8 bits of the operand address.

This makes sense from a programming standpoint because the target of a jump or subroutine jump will most often be a location in the same bank as the jump or subroutine jump itself.

Clarification of 65816 Block Move Instructions

To: Columbia Software
Dan Hillman

From: Jim Jatczynski

Date: 5 June 1985

INTRODUCTION

This memo describes the operation of the 65816 block move instructions MVP (block MoVe Positive) and MVN (block MoVe Negative). The memo is needed because the data sheet description is sketchy and the choice of mnemonics somewhat counterintuitive.

INSTRUCTION DESCRIPTION

The block move instructions, MVP and MVN, move a contiguous block from 1 to 65,536 bytes long from a source location to a destination location. Neither the source block nor destination block may straddle an interbank boundary. However, the source and destination blocks may overlap.

The instructions are three bytes long. Byte 1 is the opcode. Define **source address** as the 24-bit address of the first byte to be copied and **destination address** as the 24-bit address of the first location to be copied into. Then, byte 3 of the instruction gives the high-order 8 bits of source address, and byte 2 gives the high-order 8 bits of destination address. The X Register holds the low-order 16 bits of source address, and the Y Register holds the low-order 16 bits of destination address. The Accumulator (C Register) holds 1 less than the number of bytes to move. Contents of the X, Y, and C Registers are updated after each byte move, so the instructions are interruptable and restartable. As a side effect, the instructions also trash the contents of the Data Bank Register (DB) by loading it with the destination bank number (high-order 8 bits of destination address). Execution time is 7 cycles per byte copied.

The two instructions differ in the order in which bytes are copied. MVP starts copying at the high end (highest address) of the source and destination blocks and continues with successively lower addresses. Thus, the X and Y registers are initialized to point to the high end of the source and destination blocks; as each byte is copied, X and Y are decremented. MVN starts copying at the low end (lowest address) of the source and destination blocks and continues with successively higher addresses. Thus, the X and Y registers are initialized to point to the low end of the source and destination blocks; as each byte is copied, X and Y are incremented.

At this point, 8 out of 10 programmers probably think I've reversed the descriptions of the two mnemonics. Au contraire. If the source and destination blocks are non-overlapping, the choice between MVP and MVN depends only on whether pointers to the beginnings or ends of the blocks are more conveniently available. However, if the source and destination blocks overlap and the programmer's intention is to copy all contents of the source to the destination, a correct choice between MVP and MVN is required. If the destination is at a *higher* address than the source, use MVP to move the block as a whole in a *positive* direction. If the destination is at a *lower* address than the

source, choose MVN to move the block as a whole in a *negative* direction. The byte copying order described in the last paragraph produces the correct outcome.

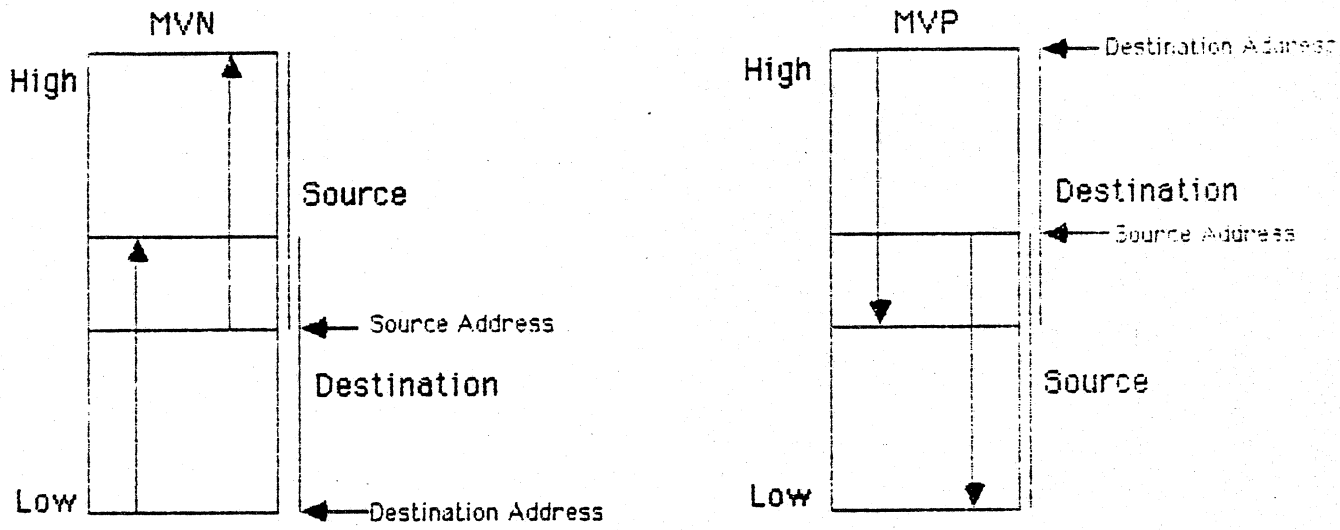
OBSERVATIONS

Speed. On a 2.8 MHz machine, MVP and MVN move each byte in 2.5µs, giving a rate of 400,000 by/s. Times required for typical block sizes are:

Size (Kby)	Time (ms)
.25	.64
.5	1.28
1	2.56
8	20.48
16	40.96
32	81.92
64	163.84

Design. These instructions provide a good example of poor instruction design. The problem is that the source and destination bank values are in the instruction itself rather than either in registers or on the stack. This makes it impossible to write a general purpose move routine using MVP and MVN without using self modifying code. The problem is a bit more difficult if the move routine has to be in ROM.

MVN, MVP Block Move Negative, Block Move Positive



Arrows indicate order in which bytes are copied.

Instruction Format

Opcode	Destination Bank	Source Bank

Instruction Setup

X Register – Low-order 16 bits of source address
Y Register – Low-order 16 bits of destination address
C Register – 1 less than the number of bytes to move

Restrictions

Source block may not straddle bank boundary
Destination block may not straddle bank boundary
Maximum block size: 65,536 bytes

Time: 7 cycles per byte moved

Effect on condition codes: None