1

第1章●データ転送命令

MOVD

MMX　SSE　SSE2　SSE3

構文

AT&Tニーモニック

```
MOVD r32/m32, mm     (MMX)
MOVD mm, r32/m32     (MMX)
MOVD r32/m32, xmm    (SSE2)
MOVD xmm, r32/m32    (SSE2)
```

Intelニーモニック

```
MOVD mm, r32/m32     (MMX)
MOVD r32/m32, mm     (MMX)
MOVD xmm, r32/m32    (SSE2)
MOVD r32/m32, xmm    (SSE2)
```

概要

MMX命令セットでサポートされるMOVDは、汎用レジスタ・MMXレジスタ間、または32ビットメモリロケーション・MMXレジスタ間で相互にダブルワードの移動を行う命令です。

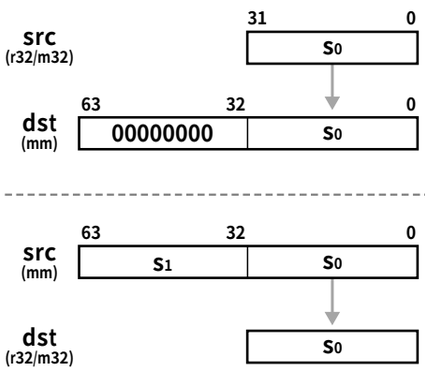

図1.1●MOVD命令（MMX）

SSE2命令セットでサポートされるMOVDは、汎用レジスタ・XMMレジスタ間、または32ビットメモリロケーション・XMMレジスタ間で相互にダブルワードの移動を行う命令です。

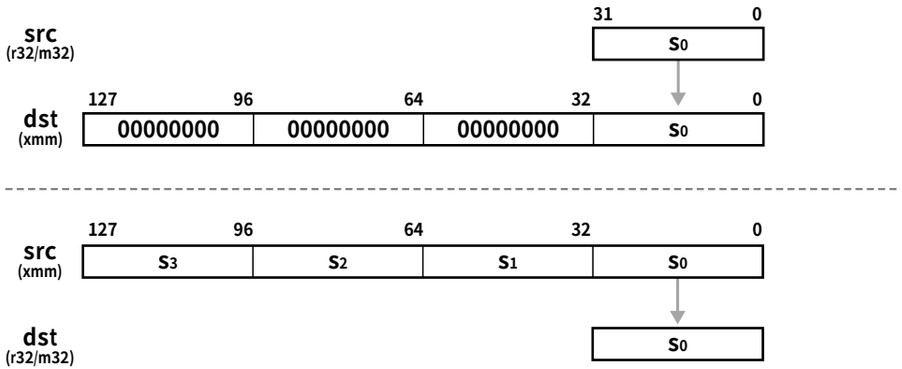

図1.2●MOVD命令（SSE2）

解説

MMX命令セットでサポートされるMOVDは、一方のオペランドとしてMMXレジスタを指定し、もう一方のオペランドには汎用レジスタ、または32ビットメモリロケーションを指定します。ディスティネーションオペランドがMMXレジスタの場合、ソースオペランドの値は下位32ビットに格納され、レジスタの上位32ビットは0クリアされます。ディスティネーションオペランドが汎用レジスタ、または32ビットメモリロケーションの場合、ソースオペランドのMMXレジスタの下位32ビットの値がディスティネーションオペランドに格納されます。

SSE2命令セットでサポートされるMOVDは、一方のオペランドとしてXMMレジスタを指定し、もう一方のオペランドには汎用レジスタ、または32ビットメモリロケーションを指定します。ディスティネーションオペランドがXMMレジスタの場合、ソースオペランドの値は下位32ビットに格納され、レジスタの上位96ビットは0クリアされます。ディスティネーションオペランドが汎用レジスタ、または32ビットメモリロケーションの場合、ソースオペランドのXMMレジスタの下位32ビットの値がディスティネーションオペランドに格納されます。

サンプル（MMX、AT&T）

MMX命令セットでサポートされるMOVDのすべてのオペランド、つまりメモリ、汎用レジスタ、MMXレジスタの三者間で値の移動を行うプログラムです。はじめにメモリからmm0レジスタへ値を読み込み、その値をメモリと汎用レジスタに格納します。

libcom.hについては導入部の記述を参照してください。

リスト1.1●movd_mmx.c

```
#include <stdio.h>
#include "libcom.h"

void
usage(char *p)
```

```
{

    fprintf(stderr, "usage: %s [arg1]\n", p);
}

int
main(int argc, char *argv[])
{
    operand                *src, *dst0, *dst1;

/*
 *      check if 1 command line argument is given
 */
    if(argc != 2) {
        usage(argv[0]);
        return -1;
    }

/*
 *      convert each command line argument to packed value
 */
    if((src = hex2operand64(argv[1], 8, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "arg1(%s): invalid value.\n", argv[1]);
        return -1;
    }
    if((dst0 = alloc_operand64(8, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "cannot allocate memory.\n");
        return -1;
    }
    if((dst1 = alloc_operand64(8, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "cannot allocate memory.\n");
        return -1;
    }

/*
 *      show the contents of operands
 */
    printf("MOVD\n     source: %s\n", operand2str(src));

/*
 *      do the calculation with using SIMD instruction
 */
    asm volatile(
        "movd   (%0),   %%mm0\n\t"
        "movd   %%mm0,  (%1)\n\t"
        "movd   %%mm0,  %%eax\n\t"
        "movl   %%eax,  (%2)\n\t"
        "emms\n"
        :
        :   "r"(src->alligned), "r"(dst0->alligned), "r"(dst1->alligned)
        :   "%eax", "%mm0"
    );

/*
 *      show the result
 */
    printf("     result: %s\n", operand2str(dst0));
    printf("     result: %s\n", operand2str(dst1));

    free_operand(src);
    free_operand(dst0);
    free_operand(dst1);

    return 0;
}
```

リスト 1.1 の実行結果

```
> ./movd_mmx 0123456789abcdef
MOVD
     source: 0123456789abcdef
     result: 0000000089abcdef
     result: 0000000089abcdef
```

サンプル（SSE2、AT&T）

SSE2 命令セット（拡張MMX 命令セットと表現される場合もある）でサポートされる MOVD のすべてのオペランド、つまりメモリ、汎用レジスタ、XMM レジスタの三者間で値の移動を行うプログラムです。はじめにメモリから xmm0 レジスタへ値を読み込み、その値をメモリと汎用レジスタに格納します。

リスト 1.2 ● movd_xmm.c

```
#include <stdio.h>
#include "libcom.h"

void
usage(char *p)
{

    fprintf(stderr, "usage: %s [arg1]\n", p);
}

int
main(int argc, char *argv[])
{
    operand                *src, *dst0, *dst1;
```

```
/*
 *      check if 1 command line argument is given
 */
    if(argc != 2) {
        usage(argv[0]);
        return -1;
    }

/*
 *      convert each command line argument to packed value
 */
    if((src = hex2operand128(argv[1], 16, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "arg1(%s): invalid value.\n", argv[1]);
        return -1;
    }
    if((dst0 = alloc_operand128(16, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "cannot allocate memory.\n");
        return -1;
    }
    if((dst1 = alloc_operand128(16, OPERAND_ALIGNED)) == NULL) {
        fprintf(stderr, "cannot allocate memory.\n");
        return -1;
    }

/*
 *      show the contents of operands
 */
    printf("MOVD\n     source: %s\n", operand2str(src));

/*
 *      do the calculation with using SIMD instruction
 */
    asm volatile(
        "movd   (%0),    %%xmm0\n\t"
        "movd   %%xmm0, (%1)\n\t"
        "movd   %%xmm0, %%eax\n\t"
        "movl   %%eax,  (%2)\n"
        :
        :   "r"(src->alligned), "r"(dst0->alligned), "r"(dst1->alligned)
        :   "%eax", "%xmm0"
    );

/*
 *      show the result
 */
    printf("     result: %s\n", operand2str(dst0));
    printf("     result: %s\n", operand2str(dst1));

    free_operand(src);
    free_operand(dst0);
    free_operand(dst1);

    return 0;
}
```


リスト 1.2 の実行結果

```
> ./movd_xmm 0123456789abcdeffedcba9876543210
MOVD
     source: 0123456789abcdeffedcba9876543210
     result: 00000000000000000000000076543210
     result: 00000000000000000000000076543210
```

サンプル（MMX、Intel）

Windows の Visual Studio でコンパイル・実行可能な、Intel ニーモニックのサンプルプログラムを示します。Windows 環境でも、Cygwin を導入すれば AT&T ニーモニックのソースコードをそのまま使用できます。しかし Visual C++ などでは、AT&T ニーモニックのソースコードがコンパイルできないため、Intel ニーモニックに書き換える必要があります。

リスト 1.3 は、MMX 命令セットでサポートされる MOVD のすべてのオペランド、つまりメモリ、汎用レジスタ、MMX レジスタの三者間で値の移動を行うプログラムです。

リスト 1.3 ● movd_mmx

```
#include <stdio.h>

int main( int argc, char *argv[] )
{
    unsigned int src[2];
    unsigned int dst0[2];
    unsigned int dst1;

    int i;
    for(i=0;i<2;i++)
        dst0[i]=0;

    src[1]=0x01234567;
    src[0]=0x89abcdef;

    __asm
    {
        movd        mm0,  src
        movd        dst0, mm0
        movd        eax,  mm0
        mov         dst1, eax

        emms
    }
```

```
    printf("     source: %08x%08x¥n", src[1],  src[0]);
    printf("     result: %08x%08x¥n", dst0[1], dst0[0]);
    printf("     result: %08x¥n", dst1);

    return 0;
}
```

リスト 1.3 の実行結果

```
C:¥>pgm
     source: 0123456789abcdef
     result: 0000000089abcdef
     result: 89abcdef
```

サンプル（SSE2、Intel）

SSE2 命令セット（拡張MMX 命令セットと表現される場合もある）でサポートされる MOVD のすべてのオペランド、つまりメモリ、汎用レジスタ、XMM レジスタの三者間で値の移動を行うプログラムです。

リスト 1.4 ● movd_xmm

```
#include <stdio.h>

int main( int argc, char *argv[] )
{
    unsigned int src[4];
    unsigned int dst0[4];
    unsigned int dst1;

    int i;
    for(i=0;i<4;i++)
        dst0[i]=0;

    src[3]=0x01234567;
    src[2]=0x89abcdef;
    src[1]=0xfedcba98;
    src[0]=0x76543210;

    __asm
    {
        movd        xmm0, src
        movd        dst0, xmm0
        movd        eax,  xmm0
        mov         dst1, eax
    }

    printf("     source: %08x%08x%08x%08x¥n", src[3],  src[2],  src[1],  src[0]);
```



```
    printf("     result: %08x%08x%08x%08x¥n", dst0[3], dst0[2], dst0[1], dst0[0]);
    printf("     result: %08x¥n", dst1);

    return 0;
}
```

リスト 1.4 の実行結果

```
C:¥>pgm
     source: 0123456789abcdeffedcba9876543210
     result: 00000000000000000000000076543210
     result: 76543210
```