Microprocessor Resources

AMD 3DNow! undocumented instructions

By Grzegorz Mazur

Introduction

Being involved in computer architecture and computer graphics, I am quite familiar with contemporary processors and graphics stuff. With my hacking attitude I frequently dig into some undocumented details of CPUs and graphics hardware and later publish some of my findings on my website. This "professional hacking" turned out into one of my favorite hobbies, and my recent discovery is simply a result of this activity.

How it all started

June 23rd was unusually cold and rainy day in Warsaw. Approaching my computer at the University in the morning I decided to free some precious space on my tiny, 1GB "working" hard disk. Half-consciously I begun to browse through deep and complicated structure of folders, trying to find something that I would never need again. I entered the folder with various 3DNow! related files downloaded from the net long time ago. While viewing one of them I noticed the names and opcodes of instructions, and suddenly I realized that not all of the names look familiar to me. I've compared the names with instructions listed in official 3DNow! documents. Clearly, there were three names not matching anything from the manual: PF2IW, PI2FW and PSWAPW.

Tracking the instructions

Judging from their names, the instructions were not unwanted artifacts. Rather they looked like something carefully designed and later abandoned. Quickly I opened the text editor and entered the these three mnemonics together with the necessary assembler directives:

```
.model  small
.586
.k3d
.code
pf2iw   mm0, mm1
pi2fd   mm0, mm1
pswapw  mm0, mm1
end
```

Then I tried to assemble the program with MASM 6.13, which I use daily for the development of hybrid (C+assembler) utilities. MASM didn't complain about the instructions, it recognized them and translated properly, showing the machine codes (suffixes), that turned out to be 1C, 0C and BB hex respectively.

Checking the functions

I quickly wrote five short assembler functions with the purpose to simply invoke the three mentioned instructions as well as documented instructions, PF2ID and PI2FD, for comparing the results with the undocumented ones. Then I wrote a simple program in C, invoking these functions with user-entered parameters and printing the results. I compiled both modules using old 16-bit Borland C and MASM 6.13, getting a simple, DOS-based, command line driven program. It all took about 15 minutes. Then I realized that I have no computer to test the stuff (the PC on my desk has IDT C6 in it). Fortunately it turned out that my colleague in the next room has K6-2 in his PC. In a few minutes I was able to run several simple tests devised to discover the exact functions of the new instructions. Given the names of instructions, their functions were not surprising:

- PF2IW is similar to PF2ID, but it returns the 16-bits results in lower halves of 32-bit words. The out-of range values are saturated at -32768 and 32767.
- PI2FW is similar to PI2FD, but it considers as its input just the lower halves of two 32-bit words, treating them as signed integers.
- PSWAPW swaps the order of 16-bit words inside 64-bit word -- the least significant 16-bit word becomes the most significant one etc..

The more formal description of the new instructions is right below.

PF2IW mmreg1, mmreg2/mem64

Opcode: 0Fh 0Fh / 1Ch

Converts packed floating-point operand to packed 16-bit integer. The instruction is similar to PF2ID, but the result qword contains two 16-bit signed integers in bits 47..32 and 15..0. Result bits 63..48 and 31..16 are cleared.

Function:

```
IF (mmreg2/mem64[31:0] >= 2^15)
```

```
    THEN mmreg1[31:0] = 7FFFh
ELSEIF (mmreg2/mem64[31:0] <= -2^15)
    THEN mmreg1[31:0] = 8000h
ELSE mmreg1[31:0] = int(mmreg2/mem64[31:0])
IF (mmreg2/mem64[63:32] >= 2^15 )
    THEN mmreg1[63:32] = 7FFFh
ELSEIF (mmreg2/mem64[63:32] <= -2^15)
    THEN mmreg1[63:32] = 8000h
ELSE mmreg1[63:32] = int(mmreg2/mem64[63:32])
```

PI2FW mmreg1, mmreg2/mem64

Opcode: 0Fh 0Fh / 0Ch

Packed 16-bit integer to floating-point conversion.

Function:

```
mmreg1[31:0] = float(mmreg2/mem64[15:0])
mmreg1[63:32] = float(mmreg2/mem64[47:32])
```

PSWAPW mmreg1, mmreg2/mem64

Opcode: 0Fh 0Fh / 0BBh

Swap 16-bit words within 64-bit MMX word.

Function:

```
mmreg1[15..0] = mmreg2/mem64[63..48]
mmreg1[31..16] = mmreg2/mem64[47..32]
mmreg1[47..32] = mmreg2/mem64[31..16]
mmreg1[63..48] = mmreg2/mem64[15..0]
```

Significance and usefulness

As usual, the undocumented instructions should not be treated too seriously -- they may disappear any time from a future product. From the reliable source somewhere on the net I got the information that the above three instructions were abandoned from 3DNow! spec because the lack of commitment from IDT and Cyrix, so I expect that IDT WinChip2 does not support them. In near future we will see if they are supported by AMD K7. Until then we can treat them just as a curiosity.

Thinking about using the undocumented instructions, I can't see any serious application for PF2ID instruction, although I believe that its designer had something in mind. The instruction returns its results in two non-adjacent 16-bit fields. It is not easy to convert the results to a more useful form. It's even easier to convert 4 floats to packed 16-bit integers using PF2ID and PACKSSDW than using PF2IW, since the MMX instructions set does not provide for packing dwords to words without signed saturation.

PI2FW in turn, can be effectively used to convert packed shorts to floats. The same task is not easy to achieve using only documented instructions, as it requires 16-bit to 32-bit signed int conversion, not available in MMX instruction set.

PSWAPW is just what it is -- it may be effectively used to reverse the order of 16-bit data in a quadword.

Software support

What's most interesting, all three instructions are supported by Microsoft's now practically free Macro assembler - MASM, versions 6.13 and 6.14, if only 3Dnow! extensions are enabled using .k3d directive.

The author is a lecturer at the Institute of Computer Science, Warsaw University of Technology (Poland). His interests include computer architecture, computer graphics and microcomputer system design. He may be contacted through his website: http://grafi.ii.pw.edu.pl/gbm

---

Back to Book and Articles