3.6 Real stuff: Streaming SIMD extensions and advanced vector extensions in x86

This section has been set as optional by your instructor.

The original MMX (*MultiMedia eXtension*) and SSE (*Streaming SIMD Extension*) instructions for the x86 included similar operations to those found in ARMv8. COD Chapter 2 (Instructions: Language of the Computer) notes that in 2001 Intel added 144 instructions to its architecture as part of SSE2, including double precision floating-point registers and operations. It included eight 64-bit registers that can be used for floating-point operands. AMD expanded the number to 16 registers, called XMM, as part of AMD64, which Intel relabeled EM64T for its use. The figure below summarizes the SSE and SSE2 instructions.

Figure 3.6.1: The SSE/SSE2 floating-point instructions of the x86 (COD Figure 3.19).

xmm means one operand is a 128-bit SSE2 register, and `{mem|xmm}` means the other operand is either in memory or it is an SSE2 register. The table uses regular expressions to show the variations of instructions. Thus, `MOV[AU]{SS|PS|SD|PD}` represents the eight instructions `MOVASS, MOVAPS, MOVASD, MOVAPD, MOVUSS, MOVUPS, MOVUSD,` and `MOVUPD`. We use square brackets `[ ]` to show single-letter alternatives: `A` means the 128-bit operand is aligned in memory; `U` means the 128-bit operand is unaligned in memory; `H` means move the high half of the 128-bit operand; and `L` means move the low half of the 128-bit operand. We use the curly brackets `{}` with a vertical bar `|` to show multiple letter variations of the basic operations: `SS` stands for *Scalar Single* precision floating point, or one 32-bit operands in a 128-bit register; `PS` stands for *Packed Single* precision floating point, or four 32-bit operands in a 128-bit register; `SD` stands for *Scalar Double* precision floating point, or one 64-bit operand in a 128-bit register; `PD` stands for *Packed Double* precision floating point, or two 64-bit operands in a 128-bit register.

Data transfer	Arithmetic	Compare
MOV[AU]{SS|PS|SD|PD} xmm, {mem|xmm}	ADD{SS|PS|SD|PD} xmm,{mem|xmm}	CMP{SS|PS|SD|PD}
	SUB{SS|PS|SD|PD} xmm,{mem|xmm}	
MOV[HL]{PS|PD} xmm, {mem|xmm}	MUL{SS|PS|SD|PD} xmm,{mem|xmm}	
	DIV{SS|PS|SD|PD} xmm,{mem|xmm}	
	SQRT{SS|PS|SD|PD} {mem|xmm}	
	MAX{SS|PS|SD|PD} {mem|xmm}	
	MIN{SS|PS|SD|PD} {mem|xmm}	

In addition to holding a single precision or double precision number in a register, Intel allows multiple floating-point operands to be packed into a single 128-bit SSE2 register: four single precision or two double precision. Thus, the 16 floating-point registers for SSE2 are actually 128 bits wide. If the operands can be arranged in memory as 128-bit aligned data, then 128-bit data transfers can load and store multiple operands per instruction. This packed floating-point format is supported by arithmetic operations that can compute simultaneously on four singles (PS) or two doubles (PD).

In 2011, Intel doubled the width of the registers again, now called YMM, with *Advanced Vector Extensions* (AVX). Thus, a single operation can now specify eight 32-bit floating-point operations or four 64-bit floating-point operations. The legacy SSE and SSE2 instructions now operate on the lower 128 bits of the YMM registers. Thus, to go from 128-bit and 256-bit operations, you prepend the letter "v" (for vector) in front of the SSE2 assembly language operations and then use the YMM register names instead of the XMM register name. For example, the SSE2 instruction to perform two 64-bit floating-point additions

```
addpd %xmm0, %xmm4
```

becomes

```
vaddpd %ymm0, %ymm4
```

which now produces four 64-bit floating-point multiplies. Intel has announced plans to widen the AVX registers to first 512 bits and later 1024 bits in later editions of the x86 architecture.

Elaboration

AVX also added three address instructions to x86. For example, ***vaddpd*** *can now specify*

```
vaddpd %ymm0, %ymm1, %ymm4    // %ymm4 = %ymm0 + %ymm1
```

instead of the standard, two address version

```
addpd %xmm0, %xmm4            // %xmm4 = %xmm4 + %xmm0
```

(Unlike LEGv8, the destination is on the right in x86.) Three addresses can reduce the number of registers and instructions needed for a computation.

PARTICIPATION ACTIVITY 3.6.1: x86 floating-point registers and operations.

1) `MULSS`, `MULPS`, `MULSD`, and `MULPD` are valid x86 instructions.
   - O True
   - O False

2) Multiple floating-point operands can be packed into a 128-bit SSE2 register.

- True
- False

3) The YMM extension was introduced to support floating-point numbers represented in a 256-bit representation.

- True
- False

Provide feedback on this section