MMX及SSE指令

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

MMX/SSE指令集的由来

- **由MMX发展而来的SSE架构**
  - ✓ **MMX指令使用8个64位寄存器MM0~MM7，借用8个80位寄存器ST(0)~ST(7)中64位尾数所占的位，可同时处理8个字节，或4个字，或2个双字，或一个64位的数据**
  - ✓ **MMX指令并没带来3D游戏性能的显著提升，故推出SSE指令，并陆续推出SSE2、SSE3、SSSE3和SSE4等采用SIMD技术的指令集，这些统称为SSE指令集**
  - ✓ **SSE指令集将80位浮点寄存器扩充到128位多媒体扩展通用寄存器XMM0~XMM7，可同时处理16个字节，或8个字，或4个双字（32位整数或单精度浮点数），或两个四字的数据**
  - ✓ **从SSE2开始，还支持128位整数运算，或同时并行处理两个64位双精度浮点数**

IA-32中通用寄存器中的编号

编号	8 位寄存器	16 位寄存器	32 位寄存器	64 位寄存器	128 位寄存器
000	AL	AX	EAX	MM0 / ST(0)	XMM0
001	CL	CX	ECX	MM1 / ST(1)	XMM1
010	DL	DX	EDX	MM2 / ST(2)	XMM2
011	BL	BX	EBX	MM3 / ST(3)	XMM3
100	AH	SP	ESP	MM4 / ST(4)	XMM4
101	CH	BP	EBP	MM5 / ST(5)	XMM5
110	DH	SI	ESI	MM6 / ST(6)	XMM6
111	BH	DI	EDI	MM7 / ST(7)	XMM7

反映了体系结构发展的轨迹，字长不断扩充，指令保持兼容

ST（0）~ ST（7）是80位，MM0 ~MM7使用其低64位

SSE指令（SIMD操作）

- 用简单的例子来比较普通指令与数据级并行指令的执行速度
  - ✓为使比较结果不受访存操作影响，下例中的运算操作数在寄存器中
  - ✓为使比较结果尽量准确，例中设置的循环次数较大: 0x4000000=2^{26}
  - ✓例子只是为了说明指令执行速度的快慢，并没有考虑结果是否溢出

以下是普通指令写的程序

```
080484f0 <dummy_add>:                                   所用时间约为22.643816s
 80484f0:  55              push  %ebp
 80484f1:  89 e5           mov  %esp, %ebp
 80484f3:  b9 00 00 00 04  mov  $0x4000000, %ecx
 80484f8:  b0 01           mov  $0x1, %al
 80484fa:   b3 00          mov   $0x0, %bl
 80484fc: 00 c3            add   %al, %bl
 80484fe: e2 fc            loop  80484fc <dummy_add+0xc>
8048500:  5d               pop           %ebp
 8048501: c3               ret
```

循环400 0000H=2^{26}次，每次只有一个数（字节）相加

SSE指令（SIMD操作）

以下是SIMD指令写的程序

所用时间约为1.411588s

```
08048510 <dummy_add_sse>:
 8048510:  55                   push  %ebp
 8048511:  b8 00 9d 04 10       mov $0x10049d00, %eax
 8048516:  89 e5                mov   %esp, %ebp
 8048518:  53                   push   %ebx
 8048519:   bb 20 9d 04 14      mov   $0x14049d20, %ebx
 804851e:  b9 00 00 40 00       mov   $0x400000, %ecx
 8048523:  66 0f 6f 00          movdqa  (%eax), %xmm0
 8048527:  66 0f 6f 0b          movdqa  (%ebx), %xmm1
 804852b: 66 0f fc c8           paddb    %xmm0, %xmm1
 804852f: e2 fa                 loop   804852b <dummy_add_sse+0x1b>
 8048531:  5b                   pop    %ebx
 8048532:  5d                   pop    %ebp
 8048533:  c3                   ret
```

SIDM指令

22.643816s/ 1.411588s ≈16.041378,与预期结果一致! SIMD指令并行执行效率高!

循环400000H=2^{22}次，每次同时有128/8=16个数（字节）相加

SSE指令（SIMD操作）

- paddb指令（操作数在两个xmm寄存器中）
  - 一条指令同时完成16个单字节数据相加
  - 类似指令paddw同时完成8个单字数据相加
  - 类似指令psubl同时完成4个双字数据相减
- movdqa指令
  - 将双四字（128位）从源操作数处移到目标操作数处
  - 用于在 XMM 寄存器与 128 位存储单元之间移入/移出双四字，或在两个 XMM 寄存器之间移动
  - 源操作数或目标操作数是存储器操作数时，操作数必须是 16 字节边界对齐，否则将发生一般保护性异常 (#GP)
- movdqu指令
  - 在未对齐的存储单元中移入/移出双四字

更多有关SSE指令集的内容请参看Intel的相关资料