SIMD

Low Level Parallel Processing

Nabeeh Jum’ah

Hamburg University

Parallel programming

To go parallel many levels support needed together

- Processor
- OS
- Compiler
- Programmer

No parallel software with any element's support lacking

Processor level parallelism

- Multiprocessor
  - Multiple chips
  - Multiple cores
  - Multi-threaded cores
- SIMD: Single instruction multiple data

Intel’s support for SIMD

- Introduced into the IA-32 architecture
- Pentium II processor family and
- Pentium processor with MMX technology
- designed to accelerate the performance of advanced media and communications applications

SIMD

- MMX
- SSE
- SSE2

…

MMX

- Processors kept backwards compatibility with all IA-32 applications and operating-system code
- Eight new 64-bit data registers
- Three new packed data types:
  - 64-bit packed byte integers (signed and unsigned)
  - 64-bit packed word integers (signed and unsigned)
  - 64-bit packed doubleword integers (signed and unsigned)
- Instructions
  - Support new data types
  - MMX state management
- MMX technology is accessible from all the IA32-architecture execution modes

Programming environment

- MMX registers
  - Process data
  - 8 64-bit registers
  - MM0 through MM7
- General-purpose registers
  - address operands
  - hold operands (for some operations)
- Memory
- Be careful when using MMX together with x87 FPU
  - Application programming
  - task switching

Access modes

- The 64-bit access mode is used for:
  - 64-bit memory accesses
  - 64-bit transfers between MMX registers
  - All pack, logical, and arithmetic instructions
  - Some unpack instructions
- The 32-bit access mode is used for:
  - 32-bit memory accesses
  - 32-bit transfer between general-purpose registers and MMX registers
  - Some unpack instructions

MMX Data Types

- 64-bit packed byte integers
- 64-bit packed word integers
- 64-bit packed doubleword integers
- 64-bit quadword

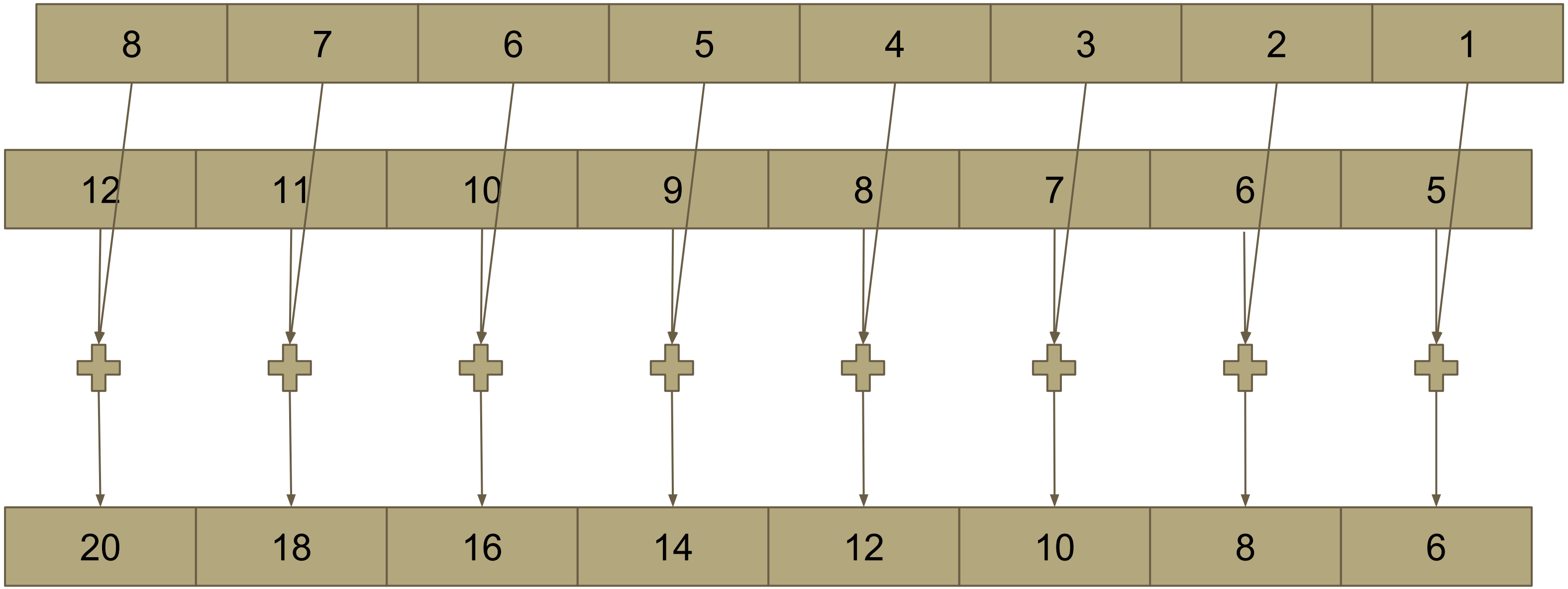
Data layout and transfer

- Little Endian
- lSB in lowest address
- MSB in highest address
- Move as a 64-bit block

Out-of-range results

- Wraparound
- Signed saturation arithmetic
- Unsigned saturation arithmetic

Operation (Example: addition)



MMX instructions

- Data transfer
- Arithmetic
- Logical
- Shift
- Comparison
- Conversion
- Unpacking
- Empty MMX state instruction (EMMS)

Data Transfer

- Transfer block size
  - MOVD
  - MOVQ
- Transfer options
  - Register to register
  - Register to memory
  - Memory to register

Arithmetic

- Addition:
  - Wraparound: PADDB, PADDW, PADDD
  - Signed Saturation: PADDSB, PADDSW
  - Unsigned Saturation: PADDUSB, PADDUSW
- Subtraction
  - Wraparound: PSUBB, PSUBW, PSUBD
  - Signed Saturation: PSUBSB, PSUBSW
  - Unsigned Saturation: PSUBUSB, PSUBUSW
- Multiplication: PMULL, PMULH
- Multiply and Add: PMADD

Logical

- Work on full quadword
- AND:
    - PAND
- AND NOT
    - PANDN
- OR
    - POR
- Exclusive OR
    - PXOR

Shift

- Packed types
  - Shift Left Logical: PSLLW, PSLLD
  - Shift Right Logical: PSRLW, PSRLD
  - Shift Right Arithmetic: PSRAW, PSRAD
- Full quadword
  - Shift Left Logical: PSLLQ
  - Shift Right Logical: PSRLQ

Comparison

- Compare for Equal
  - PCMPEQB, PCMPEQW, PCMPEQD
- Compare for Greater Than
  - PCMPGTPB, PCMPGTPW, PCMPGTPD

STREAMING SIMD EXTENSIONS (SSE)

- Introduced into the IA-32 architecture in the Pentium III processor family
- Advanced multimedia applications
  - 2-D and 3-D graphics, video, image processing, speech recognition, audio synthesis, telephony, and video conferencing
- Single-precision floating-point
  - Packed
  - Scalar
- 128 bit registers

SSE

- 128-bit data registers
  - 16 registers in 64-bit mode
  - 8 registers in other modes
  - XMM0-XMM7(..XMM15)
- The 128-bit packed single-precision floating-point data type
- Instructions to operate on packed and scalar single-precision floating-point data values
- Extended instructions on MMX packed integer values
- ...(state management, Cacheability control, prefetch, memory ordering instructions)
- In addition to MMX and GP registers, EFLAGS, ...

Data transfer

- Between registers, registers and memory
  - MOVAPS (move aligned packed single-precision floating-point values)
  - MOVUPS (move unaligned packed single-precision, floating-point)
  - MOVSS (move scalar single-precision floating-point)
  - MOVLPS (move low packed single-precision floating-point)
  - MOVHPS (move high packed single-precision floating-point)
  - MOVLHPS (move packed single-precision floating-point low to high)
  - MOVHLPS (move packed single-precision floating-point high to low)
  - MOVMSKPS (move packed single-precision floating-point mask)

Arithmetic

- Floating-Point Arithmetic
- Integer Arithmetic

Floating-Point Arithmetic

- Packed single-precision floating point
  - ADDPS, SUBPS, MULPS, DIVPS, RCPPS, SQRTPS, RSQRTPS, MAXPS, MINPS
- Scalar single-precision floating point
  - ADDSS, SUBSS, MULSS, DIVSS, RCPSS, SQRTSS, RSQRTSS, MAXSS, MINSS

Integer Arithmetic

- PAVGB, PAVGW (compute average of packed integers)
- PEXTRW, PINSRW (extract/insert word from GPR into a specific location into MMX register)
- PMAXUB, PMINUB (max/min of packed unsigned byte integers)
- PMAXSW, PMINSW (max/min of packed signed word integers)
- PMOVMSKB (move byte mask of each byte’s msb to a GPR’s low byte)
- PMULHUW (multiply packed unsigned word integers and store high result)
- PSADBW (compute sum of absolute differences)

Logical

- ANDPS
- ANDNPS
- ORPS
- XORPS

Comparison

- CMPPS
- CMPSS
- COMISS
- UCOMISS

Conversion

- Integer floating-point conversions
  - CVTPI2PS (packed doubleword integers => packed single-precision floating-point values)
  - CVTSI2SS (doubleword integer => scalar single-precision floating-point value)
  - CVTPS2PI (packed single-precision floating-point => packed doubleword integers)
  - CVTSS2SI (scalar single-precision floating-point => doubleword integer)

Other instructions

- Data processing
    - Shuffle & unpack
- Control
    - Cacheability Control Instructions
    - Prefetch
    - Store order (fence)
    - Save & restore instructions enhanced

SSE2

- Introduced in the Pentium 4 and Intel Xeon processors
- Advanced 3-D graphics, video decoding/encoding, speech recognition, E-commerce, Internet, scientific, and engineering applications
- Supported in different operating modes

SSE2

- Data types
  - 128-bit packed double-precision floating-point
  - 128-bit packed byte integers
  - 128-bit packed word integers
  - 128-bit packed doubleword integers
  - 128-bit packed quadword integers
- Instructions
  - Packed and scalar double-precision floating-point instructions
  - 128-bit packed integer instructions
  - Extended MMX and SSE instructions

Data Transfer

- Alignment
  - Aligned operations
  - Unaligned operations
- MOVAPD, MOVUPD
- MOVSD
- MOVHPD, MOVLPD
- MOVMSKPD

Arithmetic

- Double-precision floating point arithmetic
- ADDPD, SUBPD, MULPD, DIVPD
- ADDSD, SUBSD, MULSD, DIVSD
- SQRTPD, SQRTSD
- MAXPD, MINPD, MAXSD, MINSD

Logical

- ANDPD
- ANDNPD
- ORPD
- XORPD

Comparison

- CMPPD
- CMPSD
- COMISD

Conversion

- Between double-precision & single-precision floating point
  - CVTPS2PD, CVTPD2PS, CVTSS2SD, CVTSD2SS
- Between double-precision floating-point values & 32-bit integers
  - CVTPD2PI, CVTPD2DQ, CVTPI2PD, CVTDQ2PD, CVTSD2SI, CVTSI2SD
- Between single-precision floating-point values & 32-bit integers
  - CVTPS2DQ, CVTDQ2PS

Integer operations

- SSE2 64-bit & 128-bit operations
  - Transfer 128-bit integers: MOVDQA, MOVDQU
  - Arithmetic: PADDQ, PSUBQ, PMULUDQ
  - Shuffle & unpack: PSHUFLW, PSHUFHW, PSHUFD, PUNPCKHQDQ, PUNPCKLQDQ
  - 128-bit value shift logical: PSLLDQ, PSRLDQ,
  - Transfer between MMX & XMM registers: MOVQ2DQ, MOVDQ2Q
- MMX and SSE extended to 128 bit
  - All 64-bit instruction from MMX and SSE, are extended to 128-bit version besides to older 64-bit version

SIMD in x86

- More and more technologies
  - SSE3/SSSE3/SSE4/AVX ...
- More instructions/support
  - Thread synchronization, horizontal addition/subtraction, ...
- More applications
  - ex: Cryptography
- Increasing vector width
  - 256-bit, 512-bit (Knights Landing), 1024-bit? ...

- To harness processors processing power, SIMD capabilities should be best used
    - It is a powerful high-performance computing technology
- Compilers may make use of it in code optimization
- Developers may explicitly use
    - Assembly
    - Intrinsics
- It is worth, in terms of performance