Instruction Set Architecture

A few words on the subject

Grigory Rechistov

Intel–MIPT lab, MDSP project

October 2010

ISA

The part of the computer architecture related to programming.

- Instructions
- Registers
- Addressing of memory
- Native data types

ISA

The part of the computer architecture related to programming.

- Instructions
- Registers
- Addressing of memory
- Native data types

Question: why caches are not included in ISA?

ISA

The part of the computer architecture related to programming.

- Instructions
- Registers
- Addressing of memory
- Native data types

Question: why caches are not included in ISA? They are not visible to programmer.

A single instruction

[prefixes] <opcode> [operand1], [operand2], [operand3]

- Total length of an instruction may be constant or variable depending
- Prefix changes behaviour of the following operation (e.g. locked, replayed, with different length of operands)
- An operand defines what data is used and where it comes from (register, memory, constant, etc).
- Amount of operands may be even more.
- Sometimes an operand can be also placed in opcode value.
- Sometimes opcodes can be inside operand values. . .

A single instruction

[prefixes] <opcode> [operand1], [operand2], [operand3]

- Total length of an instruction may be constant or variable depending
- Prefix changes behaviour of the following operation (e.g. locked, replayed, with different length of operands)
- An operand defines what data is used and where it comes from (register, memory, constant, etc).
- Amount of operands may be even more.
- Sometimes an operand can be also placed in opcode value.
- Sometimes opcodes can be inside operand values...

Operations

Common types:

1. Set a register to constant value or value of other register.
2. Loads (memory $\rightarrow$register) & stores (register $\rightarrow$memory)
3. Read and write data from hardware devices (IO)
4. Arithmetic and Logic:
   - 4.1 $+ - /*\ldots$
   - 4.2 And, Or, Xor, Not
   - 4.3 Compare two values (result $\rightarrow$flags register)
5. Control flow
   - 5.1 branch to another location (PC $\leftarrow$new value)
   - 5.2 conditionally branch (if (condition) then PC $\leftarrow$new value)
   - 5.3 save current location and jump to new location (Procedure call)

Registers

- Program counter aka PC – the immanent part of von Neumann machines.
- Data registers: accumulators, counters.
- Address registers.
- General purpose registers (GPR): both data and address.
- Status register (flags).
- Stack pointer.

Also can include

- Segment registers.
- Coprocessor device registers e.g. FPU.
- Other types: control, debug, constant, model specific, vector registers, ...

Amount of registers on misc. architectures

Architecture	Integer registers	Double FP registers
x86	8	8
x86-64	16	16
Itanium	128	128
UltraSPARC	32	32
POWER	32	32
Alpha	32	32
6502[1]	3	0
AVR microcontroller	32	0
ARM	16	16
MMIX	256	N/A

[1] I need your clothes, boots and your motorcycle. Kill all humans! (Except Fry)

Memory addressing modes

$$\textit{Effective memory address} = f(\textit{mode}, \textit{registers}, \textit{constants})$$

There are a vast amount of modes found in different architectures, just to name some common:

- absolute
- register indirect
- PC relative
- base register + offset
- SIB: $\textit{addr} = \textit{base} + \textit{scale} * \textit{index} + \textit{offset}$

Data types

- Integer values, signed and unsigned.
- Memory adresses (mind *segment:offset* scheme in early x86)
- Boolean types: flag register, predicate registers (64 in Itanium)
- Floating point types: 64(80) bit in x87, 2×64 or 4×32 bit in SSE
- Binary coded decimal (BCD): one byte codes 100 numbers from 0x00 to 0x99.
- Tags for memory protection. Elbrus, IBM AS/400.

Data types

- Integer values, signed and unsigned.
- Memory adresses (mind *segment:offset* scheme in early x86)
- Boolean types: flag register, predicate registers (64 in Itanium)
- Floating point types: 64(80) bit in x87, 2×64 or 4×32 bit in SSE
- Binary coded decimal (BCD): one byte codes 100 numbers from 0x00 to 0x99.
- Tags for memory protection. Elbrus, IBM AS/400.

Questions:

1. What is a byte?

Data types

- Integer values, signed and unsigned.
- Memory adresses (mind *segment:offset* scheme in early x86)
- Boolean types: flag register, predicate registers (64 in Itanium)
- Floating point types: 64(80) bit in x87, 2×64 or 4×32 bit in SSE
- Binary coded decimal (BCD): one byte codes 100 numbers from 0x00 to 0x99.
- Tags for memory protection. Elbrus, IBM AS/400.

Questions:

1. What is a byte? The minimal addressable amount of data.

Data types

- Integer values, signed and unsigned.
- Memory adresses (mind *segment:offset* scheme in early x86)
- Boolean types: flag register, predicate registers (64 in Itanium)
- Floating point types: 64(80) bit in x87, 2×64 or 4×32 bit in SSE
- Binary coded decimal (BCD): one byte codes 100 numbers from 0x00 to 0x99.
- Tags for memory protection. Elbrus, IBM AS/400.

Questions:

1. What is a byte? The minimal addressable amount of data.
2. What is a machine word?

Data types

- Integer values, signed and unsigned.
- Memory adresses (mind *segment:offset* scheme in early x86)
- Boolean types: flag register, predicate registers (64 in Itanium)
- Floating point types: 64(80) bit in x87, 2×64 or 4×32 bit in SSE
- Binary coded decimal (BCD): one byte codes 100 numbers from 0x00 to 0x99.
- Tags for memory protection. Elbrus, IBM AS/400.

Questions:

1. What is a byte? The minimal addressable amount of data.
2. What is a machine word? The maximum amount of data a CPU can chew at a time.

One more time on RICS and CISC

- CISC: more close to high-level languages, compact code, variable instruction length.
- RISC: fixed length, simplier decoder, easier to pipeline, overall faster.

One more time on RICS and CISC

- CISC: more close to high-level languages, compact code, variable instruction length.
- RISC: fixed length, simplier decoder, easier to pipeline, overall faster.

Question: what architecture is IA-32?

One more time on RICS and CISC

- CISC: more close to high-level languages, compact code, variable instruction length.
- RISC: fixed length, simplier decoder, easier to pipeline, overall faster.

Question: what architecture is IA-32? CISC, alas.

Number of operands: 3

More common for CISC:

```
IMUL EAX, DWORD [EBX + ECX*4 + OFFSET], 5
BLENDPD XMM1, XMM2, 255
```

Number of operands: 2

Commonly the first operand is both the destination and source1, the second is source2:

```
ADD EAX, EBX
MOV [EBX], EBP
```

Widely used in ISAs.

Number of operands: 1

So called accumulator machines: early computers and small microcontrollers. Most instructions specify a single explicit right operand. Implicit accumulator is used as both destination and left operand.
Also, can be used with stack machines for loading/storing values from/to stack and memory:

```
PUSH A
POP RAX
```

Number of operands: Zero

Number of operands: Zero

0 ? WTF?

Number of operands: Zero

0 ? WTF?
So called stack machines.
All arithmetic operations take place using the top one or two positions on the stack.
Esoteric stuff.

A note on microarchitecture

Microarchitecture $\neq$ ISA

- IBM TIMI (Technology-Independent Machine Interface): IBM AS/400 (48 bit CISC)$\rightarrow$ POWER (RISC 64 bit)
- Transmeta: x86-compatible ISA on VLIW μarch.
- Think about Intel vs AMD internal CPU designs.

Instruction set extensions

Are made when somebody decides it's better to have some new instructions.

1. 80286 – 16 bit;
2. 80386 – 32 bit;
3. coprocessor 8087;
4. MMX, SSEx, AVX extensions for SIMD operations;
5. AMD part: 3DNow! and AMD64 aka EMT-64 aka x86_64.
6. Virtualization support.

Instruction set extensions

Are made when somebody decides it's better to have some new instructions.

1. 80286 – 16 bit;
2. 80386 – 32 bit;
3. coprocessor 8087;
4. MMX, SSEx, AVX extensions for SIMD operations;
5. AMD part: 3DNow! and AMD64 aka EMT-64 aka x86_64.
6. Virtualization support.

Retain backward compatibility! Remember the lessons of Itanium.

Some examples of ISAs

Some examples of ISAs

- Alpha

Some examples of ISAs

- Alpha
- ARM

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson
- MIPS

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson
- MIPS
- Motorola 68k

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson
- MIPS
- Motorola 68k
- PowerPC

Some examples of ISAs

- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson
- MIPS
- Motorola 68k
- PowerPC
- SPARC

Some examples of ISAs

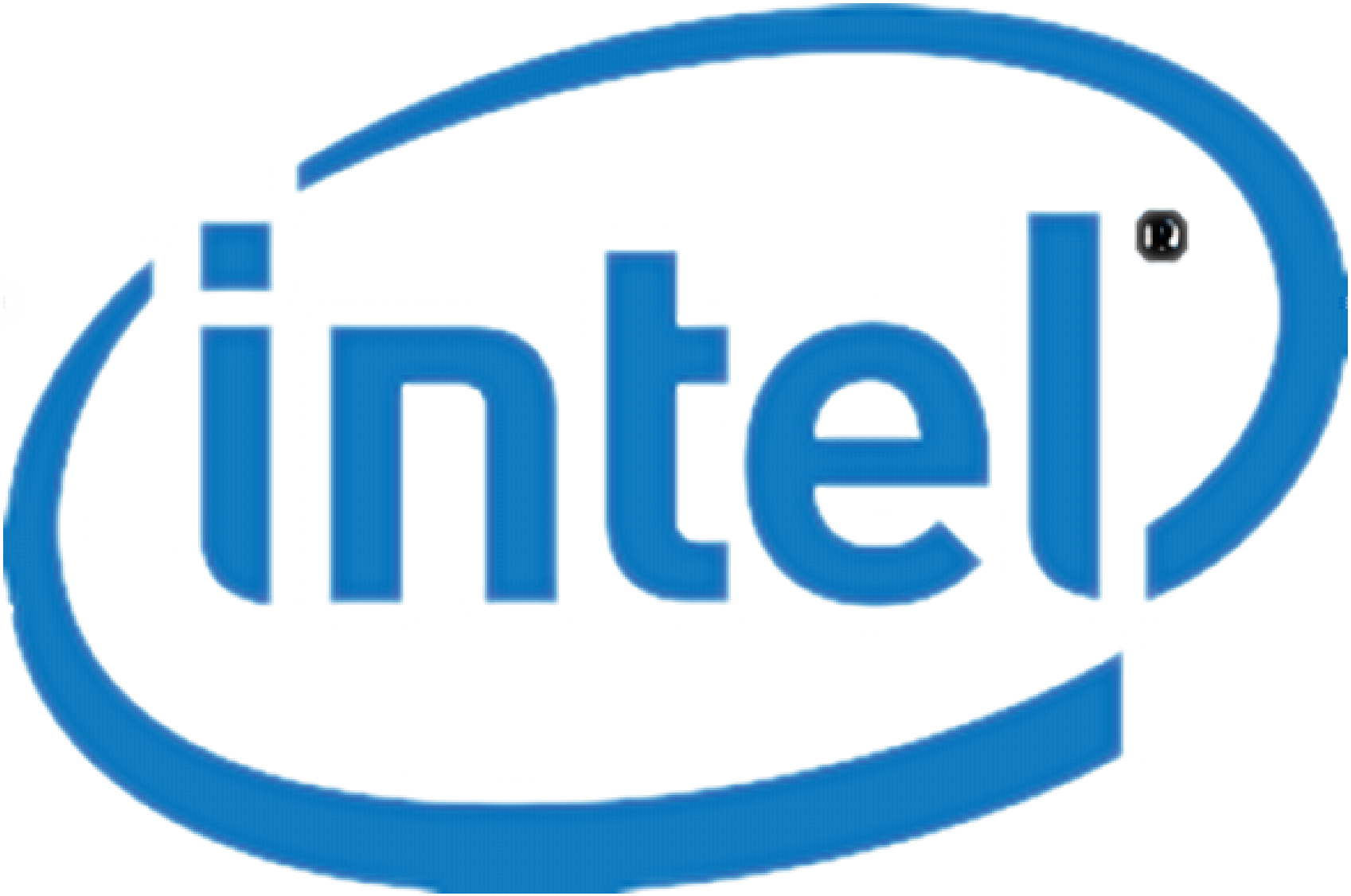
- Alpha
- ARM
- IA-64 (Itanium)
- IA-32, including EMT64 aka AMD64
- Loongson
- MIPS
- Motorola 68k
- PowerPC
- SPARC

ISAs commonly implemented in software with hardware incarnations

- Java virtual machine (ARM Jazelle)
- FORTH
- MMIX from *The Art of Computer Programming*

More to read

- Wikipedia
  http://en.wikipedia.org/wiki/Instruction_set
- The Art of Assembly Language, chapter 5.
  http://webster.cs.ucr.edu/AoA/Windows/HTML/ISA.html


intel