Compiler Construction

Microprocessors

Problem statement

Quick Overview of microprocessors

Choose one microprocessor for this course

$\Rightarrow$ Easy to understand
$\Rightarrow$ Fit modern architectures
$\Rightarrow$ Correspond to most of the actual compilers

Instruction set

> “ *Instruction set architecture is the structure of a computer that a machine language programmer (or a compiler) must understand to write a correct (timing independent) program for that machine*
>
> —
>
> *IBM introducing 360 (1964)*

The Instruction Set Architecture (ISA) is the part of the processor that is visible to the programmer or compiler writer.

What is an instruction set?

An instruction set specifies a processor functionality:

- what operations are supported
- what storage mechanisms are used
- how to access storage
- how to communicate between program and processor

Technical aspect of instruction set

1. format: length, encoding
2. operations: data type (floating or fixed point) , number and kind of operands
3. storage:
   - internal: accumulator, stack, register
   - memory: address size, addressing modes
4. control: branch condition, support for procedures, predication

What makes a good instruction set?

An instruction set specifies a processor functionnality:

- **implementability**: support for a (high performance) range of implementation
- **programmability**: easy to express program (by Humans before 80's, mostly by compiler nowadays)
- **backward/forward compatibility**: implementability & programmability across generations

CISC – Complex Instruction Set Chip

- large number of instructions (100-250)
- 6, 8, 16 registers, some for pointers, others for integer computation
- arithmetic in memory can be processed
- two address code
- many possible effects (e.g., self-incrementation)

CISC – Pros & Cons

Pros:

- Simplified compiler
- Small assembly code
- Few instructions will be fetched
- Special purpose register available: stack pointer, interrupt handling ...

Cons:

- Variable length instruction format
- Variable number of clock cycles per instruction
- Limiter number of general purpose registers

Motivations for something else!

Though the CISC programs could be small in length, but number of bits of memory occupies may not be less

The complex instructions do not simplify the compilers: many clock cycles can be wasted to find the appropriate instruction.

RISC architectures were designed with the goal of executing one instruction per clock cycle.

RISC – Reduced Instruction Set Chip

- 32 generic purpose registers
- arithmetic only available on registers
- 3 address code
- `load` and `store` relative to a register (`M[r + const]`)
- only one effect or result per instruction

RISC – Pipeline 1/3

Pipelining is the overlapping the execution of several instructions in a pipeline fashion.

A pipeline is (typically) decomposed into five stages:

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execute (EX)
4. Memory Access (MA)
5. Write Back (WB)

RISC – Pipeline 2/3

inst1:	IF	ID	EX	MA	WB				
inst2:		IF	ID	EX	MA	WB			
inst3:			IF	ID	EX	MA	WB		
inst4:				IF	ID	EX	MA	WB	
inst5:					IF	ID	EX	MA	WB

The slowest stage determines the speed of the whole pipeline!

Ex introduces latency

- Register-Register Operation: 1 cycle
- Memory Reference: 2 cycles
- Multi-cycle Instructions (floating point): many cycles

RISC – Pipeline 3/3

Data hazard: When an instruction depends on the results of a previous instruction still in the pipeline.

- inst1 write in $s1 during WB
- inst1 read in $s1 during ID

inst1:	IF	ID	EX	MA	WB	
inst2:		IF	ID	EX	MA	WB

inst2 must be split, causing delays...

other dependencies can appear

RISC – Pros & Cons

Pros:

- Fixed length instructions: decoding is easier
- Simpler hardware: higher clock rate
- Efficient Instruction pipeline
- Large number of general purpose registers
- Overlapped register windows to speed up procedure call and return
- One instruction per cycle

Cons:

- Minimal number of addressing modes: only Load and Store
- Relatively few instructions

Nowadays

- the classification pure-RISC or pure-CISC is becoming more and more inappropriate and may be irrelevant
- modern processors use a calculated combination elements of both design styles
- Some processors that are classified as CISC while employing a number of RISC features, such as pipelining

ARM provides the advantage of using a CISC (in terms of functionality) and the advantage of an RISC (in terms of reduced code lengths).

Lessons to be learned

Implementability

Driven by technology: microcode, VLSI, FPGA, pipelining, superscalar, SIMD, SSE

Programmability

Driven by compiler technology

Sum-up

- Many non technical issues influence ISA's
- Best solutions don't always win (Intel X86)

Intel X86 (IA32)

- Introduced in 1978
- 8×32 bits ”general” register
- variable length instructions (1–15 byte)
- long life to the king!

Intel’s trick?

Decoder translates CISC into RISC micro-operations

Summary