Advanced Computer Architecture

(0630561)

Lecture 2

RISC Architecture

Prof. Kasim M. Al-Aubidy

Computer Eng. Dept.

Reduced Instruction Set Computer (RISC):

- RISC architectures represent an important innovation in the area of computer organization.
- The RISC architecture is an attempt to produce more CPU power by simplifying the instruction set of the CPU.
- The opposed trend to RISC is that of complex instruction set computers (CISC).
- Both RISCs and CISCs try to solve the same problem. CISCs are going the traditional way of implementing more and more complex instructions. RISCs try to simplify the instruction set.
- Innovations in RISC architectures are based on a close analysis of a large set of widely used programs.
- One of the main concerns of RISC designers was to maximize the efficiency of pipelining.
- Present architectures often include both RISC and CISC features.
- **Both RISC and CISC architectures have been developed as an attempt to cover the semantic gap.**

- In order to improve the efficiency of software development, new and powerful programming languages have been developed (Ada, C++, Java). They provide: high level of abstraction, conciseness, power.

Problem: How should new HLL programs be compiled and executed efficiently on a processor architecture?

Two possible answers:

1. The CISC approach: design very complex architectures including a large number of instructions and addressing modes; include also instructions close to those present in HLL.
2. The RISC approach: simplify the instruction set and adapt it to the real requirements of user programs.

Are RISCs Really Better than CISCs?

- RISC architectures have several advantages and they were discussed throughout this lecture. However, a definitive answer to the above question is difficult to give.
- A lot of performance comparisons have shown that benchmark programs are really running faster on RISC processors than on processors with CISC characteristics.
- However, it is difficult to identify which feature of a processor produces the higher performance. Some "CISC fans" argue that the higher speed is not produced by the typical RISC features but because of technology, better compilers, etc.
- An argument in favour of the CISC: the simpler instruction set of RISC processors results in a larger memory requirement compared to the similar program compiled for a CISC architecture.
- **Most of the current processors are not typical RISCs or CISCs but try to combine advantages of both approaches**

Some Processor Examples

CISC Architectures:

	VAX 11/	Pentium
Nr. of instructions:	303	235
Instruction size:	2 – 57	1-11
Instruction format:	not fixed	not fixed
Addressing modes:	22	11
Number of GP registers:	16	8

RISC Architectures:

	Sun SPARC	PowerPC
Nr. of instructions:	52	206
Instruction size:	4	4
Instruction format:	fixed	not fixed
Addressing modes:	2	2
Number of GP registers:	up to 520	32

Characteristics	CISC Machines			RISC Machines	
Processor	IBM370	VAX	i486	SPARC	MIPS
Year developed	1973	1978	1989	1987	1991
No. of instructions	208	303	235	69	94
Inst. size (bytes)	2-6	2-57	1-12	4	4
Addressing modes	4	22	11	1	1
No. of G.P. registers	16	16	8	40-520	32
Cont. M. size (Kbits)	420	480	246	-	-

Main Characteristics of RISC Architectures:

- The instruction set is limited and includes only simple instructions.
- Only LOAD and STORE instructions reference data in memory.
- Instructions use only few addressing modes.
- Instructions are of fixed length and uniform format.
- A large number of registers is available.

Main Characteristics of RISC

- A small number of simple instructions
  - Simple and small decode and execution hardware are required.
  - A hard-wired controller is needed, rather than using microprogramming.
  - The CPU takes less silicon area to implement, and runs also faster.
- Main characteristics
  - Execution of one machine instruction per clock cycle
  - Register-to-register operations
  - Simple addressing modes
  - Simple instruction formats

One Instruction per Clock Cycle

- Machine cycle is defined to be the time it takes to fetch 2 operands from registers, perform an ALU operation and store the result in a register.
- The instruction pipeline performs more efficiently due to simple instructions and similar execution patterns.
- Complex operations are executed as a sequence of simple instructions.
  - In the case of CISC they are executed as one single or a few complex instruction.

Example:

An illustrative example with the following assumption:

- A program with 80% of executed instructions being simple and 20% complex.
- CISC: simple instructions take 4 cycles, complex instructions take 8 cycles; cycle time is 100 ns.
- RISC: simple instructions are executed in one cycle; complex operations are implemented as a sequence of instructions (14 instructions on average); cycle time is 75 ns.

How much time takes a program of 1 000 000 instructions?

- CISC: $(10^6 \times 0.80 \times 4 + 10^6 \times 0.20 \times 8) \times 10^{-7} = 0.48$ s
- RISC: $(10^6 \times 0.80 \times 1 + 10^6 \times 0.20 \times 14) \times 0.75 \times 10^{-7} = 0.27$ s

Register to Register Operation

- **Load-and-store architecture**
  - Only LOAD and STORE instructions reference data in memory.
  - All other instructions operate only with registers
- This characteristic simplifies the instruction set and therefore the control unit.
  - A RISC instruction set may include only 1 or 2 ADD instructions (e.g. integer add, add with carry)
  - VAX has 25 different ADD instructions
- This architecture encourages the optimization of register use, so that frequently accessed operands remain in high-speed storage.

Register to Register Operation

A large number of registers is available.

- Variables and intermediate results can be stored in registers and do not require repeated loads and stores from/to memory.
- All local variables of procedures and the passed parameters can be stored in registers.
- The large number of registers is typical for RISC, because the reduced complexity of the processor means that we have silicon space on the processor chip to implement them. This is usually not the case with CISC machines.

Main Characteristics of RISC:

- Only a few simple addressing modes are used.
  - Almost all RISC instructions use simple register addressing
  - Complex modes can be synthesized in software from the simple ones
  - Ex. register, direct, register indirect, displacement.

- Instructions are of fixed length and uniform format.
  - Loading and decoding of instructions are simple and fast; it is not needed to wait until the length of an instruction is known in order to start decoding it;
  - Decoding is simplified because the opcode and address fields are located in the same position for all instructions.

Main Advantages of RISC:

- Best support is given by optimizing most used and most time consuming architecture aspects.
  - Frequently executed instructions.
  - Memory reference.
  - Procedure call/return.
  - Pipeline design.
- Less design complexity, reducing design cost, and reducing the time between designing and marketing.

Criticism of RISC:

- An operation might need two, three, or more instructions to accomplish.
  - More memory access might be needed.
  - Execution speed may be reduced in certain applications.
- It usually leads to longer programs, which needs larger memory space to store.
- Difficult to program **m c codes** and **assembly programs**.
  - More time consuming.

Characteristics of Some Processors

	Processor	Number of instruction sizes	Max instruction size in bytes	Number of addressing modes	Indirect addressing	Load/store combined with arithmetic	Max number of memory operands	Unaligned addressing allowed	Max Number of MMU uses	Number of bits for integer register specifier	Number of bits for FP register specifier
RISC	AMD29000	1	4	1	no	no	1	no	1	8	3[a]
	MIPS R2000	1	4	1	no	no	1	no	1	5	4
	SPARC	1	4	2	no	no	1	no	1	5	4
	MC88000	1	4	3	no	no	1	no	1	5	4
	HP PA	1	4	10[f]	no	no	1	no	1	5	4
	IBM RT/PC	2[a]	4	1	no	no	1	no	1	4[g]	3[a]
	IBM RS/6000	1	4	4	no	no	1	yes	1	5	5
	Intel i860	1	4	4	no	no	1	no	1	5	4
CISC	IBM 3090	4	8	2[b]	no[b]	yes	2	yes	4	4	2
	Intel 80486	12	12	15	no[b]	yes	2	yes	4	3	3
	NSC 32016	21	21	23	yes	yes	2	yes	4	3	3
	MC68040	11	22	44	yes	yes	2	yes	8	4	3
	VAX	56	56	22	yes	yes	6	yes	24	4	0
	Clipper	4[e]	8[e]	9[e]	no	no	1	0	2	4[e]	3[e]
	Intel 80960	2[e]	8[e]	9[e]	no	no	1	yes[e]	–	5	3[e]

Main features of CISC

- A large number of instructions (> 200) and complex instructions and data types.
- Many and complex addressing modes.
- Direct hardware implementations of high-level language statements.
- Microprogramming techniques are used so that complicated instructions can be implemented.
- Memory bottleneck is a major problem, due to complex addressing modes and multiple memory accesses per instruction.

Arguments for CISC

- A rich instruction set should simplify the compiler by having instructions which match the high-level language instructions.
- Since the programs are smaller in size, they have better performance:
  - They take up less memory space.
  - Fewer number of instructions are executed and need fewer instruction fetch cycles, which may lead to smaller execution time.
  - In a paging environment, smaller programs occupy fewer pages, reducing page faults.
- Program execution efficiency is also improved by implementing complex operations in microcode rather than machine code.

Problems with CISC

- Compiler simplification?
  - Disputed...
  - Complex machine instructions harder to exploit
  - Optimization more difficult
- Smaller programs?
  - Program takes up less memory but...
  - Memory is now cheap
  - May not occupy less bits, just look shorter in symbolic form
    - More instructions require longer op-codes
    - Register references require fewer bits
- Faster programs?
  - Bias towards use of simpler instructions
  - More complex control unit
  - Microprogram control store larger
  - thus simple instructions take longer to execute
- It is far from clear that CISC is the appropriate solution

Problems with CISC

- A large instruction set requires complex and potentially time consuming hardware steps to decode and execute the instructions.
- Complex machine instructions may not match high-level language statements exactly, in which case they may be of little use.
  - This will be a major problem if the number of languages is getting bigger.
- Instruction sets designed with specialized instructions for several high-level languages will not be efficient when executing program of a given language.
- Complex design tasks.

CISC	RISC
1. Large number of instructions – from 120 to 350.	1. Relatively fewer instructions - less than 100.
2. Employs a variety of data types and a large number of addressing modes.	2. Relatively fewer addressing modes.
3. Variable-length instruction formats.	3. Fixed-length instructions usually 32 bits, easy to decode instruction format.
4. Instructions manipulate operands residing in memory.	4. Mostly register-register operations. The only memory access is through explicit LOAD/STORE instructions.
5. Number of Cycles Per Instruction (CPI) varies from 1-20 depending upon the instruction.	5. Number of CPI is one as it uses pipelining. Pipeline in RISC is optimised because of simple instructions and instruction formats.
6. GPRs varies from 8-32. But no support is available for the parameter passing and function calls.	6. Large number of GPRs are available that are primarily used as Global registers and as a register based procedural call and parameter passing stack, thus, optimised for structured programming.
7. Microprogrammed Control Unit.	7. Hardwired Control Unit.

A CISC Example – Intel i486

- 32-bit integer processor
  - Registers 8 general 2 status/control
    6 address (16-bits) 1 instruction-pointer (PC)
  - On-chip floating-point unit
  - Microprogrammed control
- Instruction set:
  - Number of instructions: 235
  - Instruction size: 1-12 bytes (3.2 on average).
  - Addressing modes: 11
- Memory organization:
  - Address length: 32 bits (4 GB space)
  - Memory is segmented for protection purpose
  - Support virtual memory by paging
  - Cache-memory: 8 KB internal (on-chip)
- Instruction execution:
  - Execution models: reg-reg, reg-mem, mem-mem
  - Five-stage instruction pipeline: Fetch, Decode1, Decode2, Execute, Write-Back.

A RISC Example – SPARC

Scalable Processor Architecture:

- 32-bit processor with the following components:
  - An Integer Unit (IU) — to execute all instructions.
  - A Floating-Point Unit (FPU) — a co-processor which can work concurrently with the IU and is used to perform arithmetic operations on floating point numbers.
- has 69 basic instructions.
- has a linear, 32-bit virtual-address space (4G).
- has 40 to 520 general purpose 32-bit registers which are grouped into 2 to 32 overlapping register windows
  (16/group + 8) => Scalability.

http://www.cse.hcmut.edu.vn/~anhvu/teaching/2010/ACA/lectures/ACA-chapter4.pdf