Lecture 4: RISC Computers

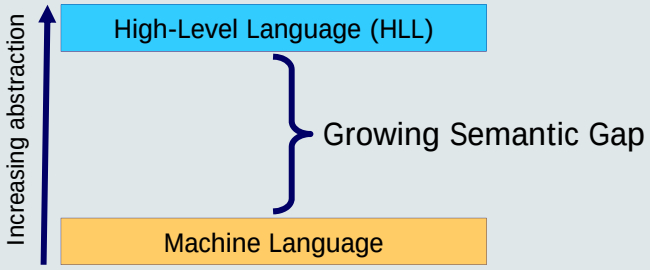
- Introduction
- Program execution features
- RISC characteristics
- RISC vs. CICS

Introduction

- **Reduced Instruction Set Computer** (RISC) represents an important innovation in computer architecture.
- It 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 Computer** (CISC).
- Both RISC and CISC architectures have been developed as an attempt to cover the **semantic gap**, and consequently to reduce the ever growing software costs.

The Semantic Gap

- In order to improve efficiency of software development, powerful high-level programming languages have been developed (e.g., Ada, C++, Java).
  - They support higher levels of abstraction.
- This evolution has increased the semantic gap between programming languages and machine languages.



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.
  - For example, CASE (switch) on VAX
- 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.

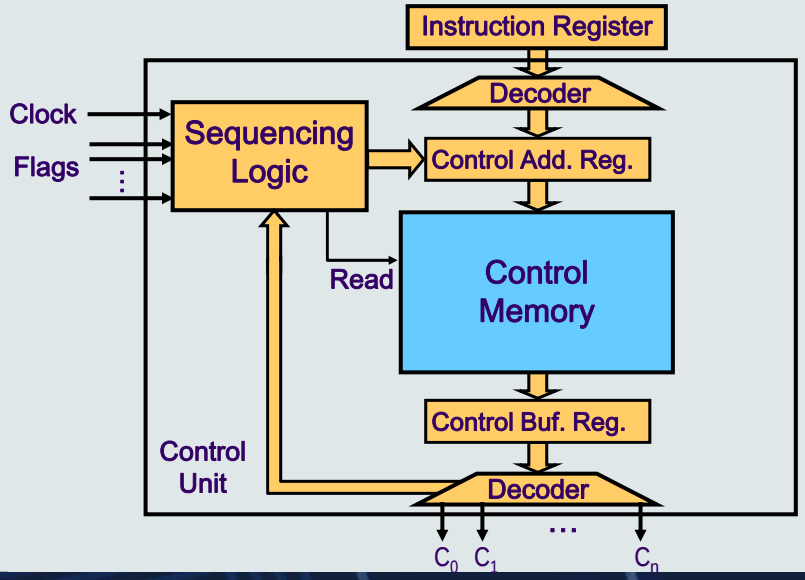
Arguments for CISC

- A rich instruction set should simplify the compiler by having instructions which match the high-level language statements.
  - This works fine if the number of HL languages is very small.
- Since the programs are smaller in size, they have better performance:
  - They take up less memory space and need fewer instruction fetch cycles.
  - Fewer number of instructions are executed, which may lead to smaller execution time.
- Program execution efficiency is also improved by implementing complex operations in microcode rather than machine code.

Microprogrammed Control

- Microprogramming is a technique used to implement the control unit.
- The basic idea is to implement the control unit as a micro-program execution machine (a computer inside a computer).
  - The set of micro-operations occurring at one time defines a microinstruction.
  - A sequence of microinstructions is called a microprogram.
- The execution of a machine instruction becomes the execution of a sequence of micro-instructions.
  - This is similar to that a C++ statement is implemented by a sequence of machine instructions.

Hardware Implementation



Microcode Storage and Execution

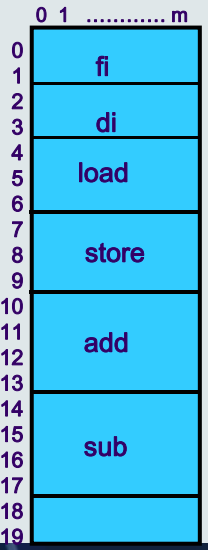
- Microcodes are stored in a micro-memory which is much faster than a cache.
- Since the micro-memory stores only programs, a ROM is often used.
- Microprograms are sometimes called firmware.

- Execution of a complex instruction in microcode, COM:

  fi → di → load → load → add → sub → store

- Execution in terms of a sequence of machine code:

  LOAD → LOAD → ADD → SUB → STORE



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.

Lecture 4: RISC Computers

- Introduction
- Program execution features
- RISC characteristics
- RISC vs. CICS

Evaluation of Program Execution

- What are programs doing most of the time?
- We need to understand execution characteristics of machine instruction sequences generated from HLL programs.
- Aspects of interest:
  - The frequency of operations performed.
  - The types of operands and their frequency of use.
  - Control flow frequency:
    - branches,
    - loops,
    - subprogram calls.

Evaluation Results

- Frequency of machine instructions executed:
  - moves: 33%
  - conditional branches: 20%
  - arithmetic/logic operations: 16%
  - others: between 0.1% and 10%
- Addressing modes:
  - the majority of instructions uses simple addressing modes.
  - complex addressing modes (memory indirect, indexed+indirect, etc.) are only used by ~18% of the instructions.

Evaluation Results (Cont'd)

- Operand types:
  - 74-80% of the operands are scalars (integers, reals, characters, etc.).
  - the rest (20-26%) are arrays/structures; 90% of them are *global variables*.
  - about 80% of the scalars are *local variables*.

Conclusion: the majority of operands are local variables of scalar type, which can be stored in registers.

Procedure Calls

Studies has also been done on the percentage of the execution time spent executing different high-level language (HLL) statements.

- Most of the time is spent executing CALLs and RETURNs in HLL programs.
- Even if only 15% of the executed HLL statements is a CALL or RETURN, they are executed most of the time, because of their complexity.
- A CALL or RETURN is compiled into a relatively long sequence of machine instructions with a lot of memory references.

Evaluation Conclusions

- An overwhelming dominance of simple (ALU and move) operations over complex operations.
- Dominance of simple addressing modes.
- Large frequency of operand accesses; on average each instruction references 1.9 operands.
- Most of the referenced operands are scalars (can be stored in registers) and are local variables or parameters.
- Optimizing the procedure CALL/RETURN mechanism promises large benefits in speed.

⇒ The RISC computers are developed to take advantages of the above results, and thus delivering better performance!

Lecture 4: RISC Computers

- **Introduction**
- **Program execution features**
- **RISC characteristics**
- **RISC vs. CICS**

Main Characteristics of RISC

- A small number of simple instructions (desirably < 100).
  - 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.
- Execution of one instruction per clock cycle.
- 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 instructions.

Main Characteristics of RISC (Cont'd)

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

Main Characteristics of RISC (Cont'd)

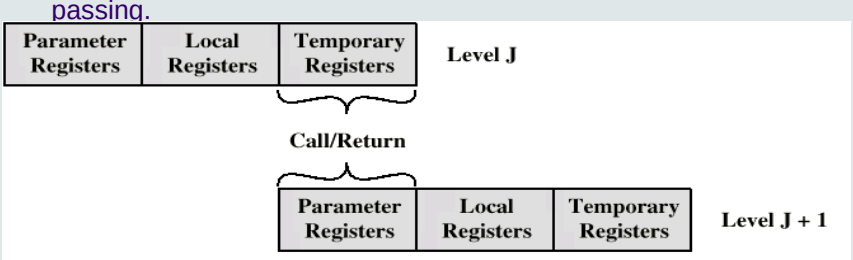
- Load-and-store architecture
  - Only LOAD and STORE instructions reference data in memory.
  - All other instructions operate only with registers (are register-to-register instructions).
- Only a few simple addressing modes are used.
  - Ex. register, direct, register indirect, displacement.
- Instructions 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 Characteristics of RISC (Cont'd)

- 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 due to that the reduced complexity of the processor leaves silicon space on the chip to implement them.
  - This is usually not the case with CISC machines.

Register Windows

- A large number of registers is usually very useful.
- However, if contents of all registers must be saved at every procedure call, more registers mean longer delay.
- A solution to this problem is to divide the register file into a set of fixed-size windows.
  - Each window is assigned to a procedure.
  - Windows for adjacent procedures are overlapped to allow parameter passing.



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 **machine codes** and **assembly programs**.
  - More time consuming.

Lecture 4: RISC Computers

- **Introduction**
- **Program execution features**
- **RISC characteristics**
- **RISC vs. CICS**

RISC vs. CISC

- Studies have shown that benchmark programs run often faster on RISC processors than on CISC machines.
- However, it is difficult to identify which RISC feature really 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 favor of the CISC: the simpler RISC instruction set results in a larger memory requirement compared with the CISC case.

⇒ Most recent processors are not typical RISC or CISC, but combine advantages of both approaches.

- e.g. PowerPC and Pentium II.

CISC/RISC Examples

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	-	-

A CISC Example — Intel i486

- 32-bit integer processor
  - Registers 8 general
    6 address (16-bits)
    2 status/control
    1 instruction-pointer (PC)
  - On-chip floating-point unit
  - Microprogrammed control
- Instruction set:
  - Number of instructions: 235
    - E.g., “FYL2XP1 x y” performs $y \bullet \log_2(x+1)$ in 313 clock cycles.
  - Instruction size: 1-12 bytes (3.2 on average).
  - Addressing modes: 11

A CISC Example — Intel i486 (Cont’d)

- 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)
    (96% hit rate for DOS applications)
- 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 global) => Scalability.

Summary

- Both RISCs and CISCs try to reduce the semantic gap, with different approaches.
  - CISCs follow the traditional way of implementing more and more complex instructions.
  - RISCs try to simplify the instruction set, while improving the instruction execution performance.
- Innovations in RISC architectures are based on a close analysis of typical programs.
- The main features of RISC architectures are:
  - reduced number of simple instructions, few addressing modes;
  - instructions with fixed length and format, load-store architecture;
  - a large number of registers.
- RISCs support efficient implementation of pipelining.
- Most modern architectures often include both RISC and CISC features.