



EE4801 – Lecture 18

Characteristics of Programs Introduction to RISC



CISC Characteristics



- ◆ CISC – Complex Instruction Set Computer
 - Typically multiple clocks per instruction cycle.
 - Complex addressing modes associated with many different instructions.
 - Non-uniform, non-orthogonal instruction definitions.
 - Relatively small set of internal registers.

RISC Characteristics

- ◆ RISC – Reduced Instruction Set Computer
 - Most instructions execute in exactly one clock cycle.
 - Very few (often only two) instructions access memory – sometimes called a “load-store” architecture.
 - Highly uniform instruction definitions.
 - Large number of internal registers.
 - Pipelined, hard-wired, instruction processing.

Guiding Principles of RISC

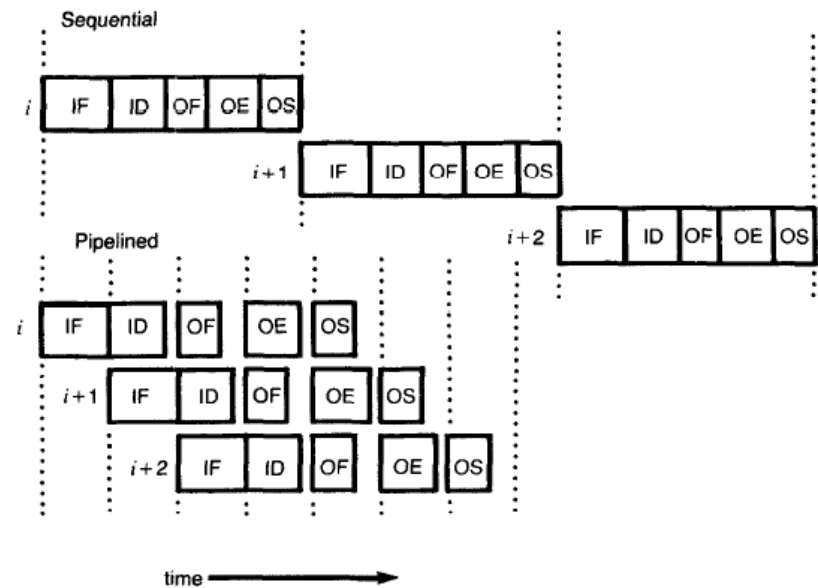
- ◆ In his seminal 1985 paper Patterson presented what became the guiding principles of RISC design:
 - Functions should be kept simple unless there is a very good reason to do otherwise.
 - Microinstructions should not be faster than simple instructions.
 - Microcode is not magic – anything that can be done in a microcoded machine can be done in assembly language in a simple machine.
 - Simple decoding and pipelined execution are more important than program size.
 - Compiler technology should be used to simplify instructions rather than to generate complex instructions.

RISC Systems

- ◆ These guiding principles led to a philosophy of design that resulted in RISC machines having many common features:
 - Operations are register-to-register, with only LOAD and STORE accessing memory.
 - The operations and addressing modes are reduced – Operations between registers complete in one cycle.
 - Instruction formats are simple and do not cross word boundaries.
 - RISC branches avoid pipeline penalties.

Advantage of Pipelining

- ◆ As we saw yesterday, instruction processing was broken in stages:
 - IF – Instruction Fetch
 - ID – Instruction Decode
 - OF – Operand Fetch
 - OE – Operand Decode
 - OS – Operand Store



Effect of Data Dependency

- ◆ A pipeline improves performance most when there are no dependencies between instructions.
- ◆ Consider the execution of the following sequence of instructions:

ADD b, c, a ; a = b+c

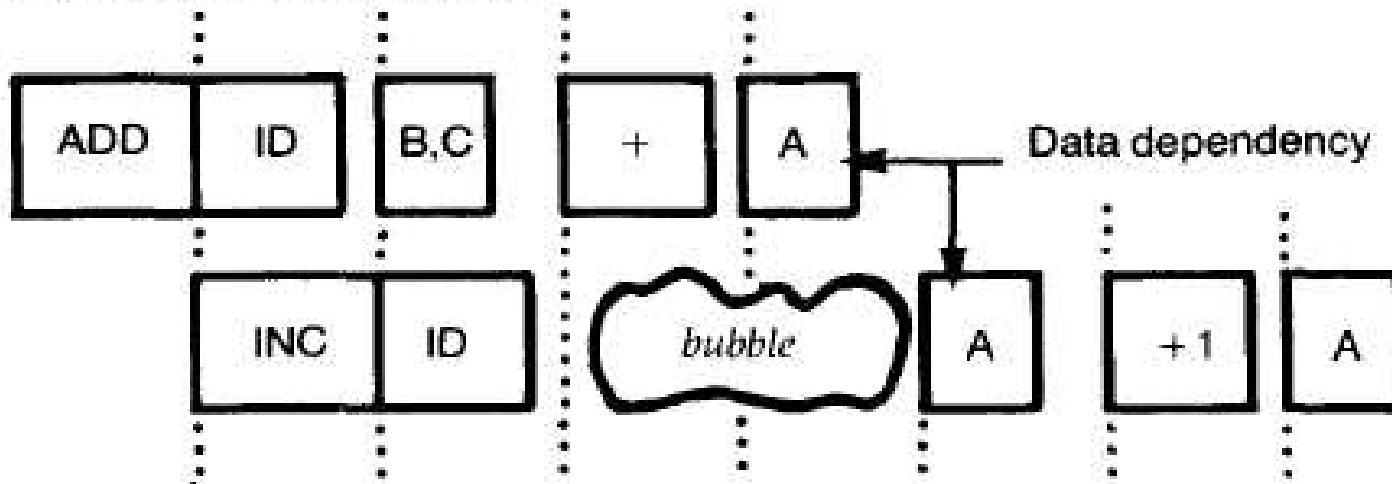
INC a ; a = a+1

- ◆ In this case, a cannot be incremented until the add a,b,c instruction completes!

Handling Data Dependency

- ◆ This situation (potentially) introduces a “bubble” into the pipeline:

Pipelined with Data Interlock



Importance of the Compiler

- ◆ Inserting a bubble reduces performance.
- ◆ Now, consider the following two code examples:

ADD b, c, a

INC a

INC b

INC c



ADD b, c, a

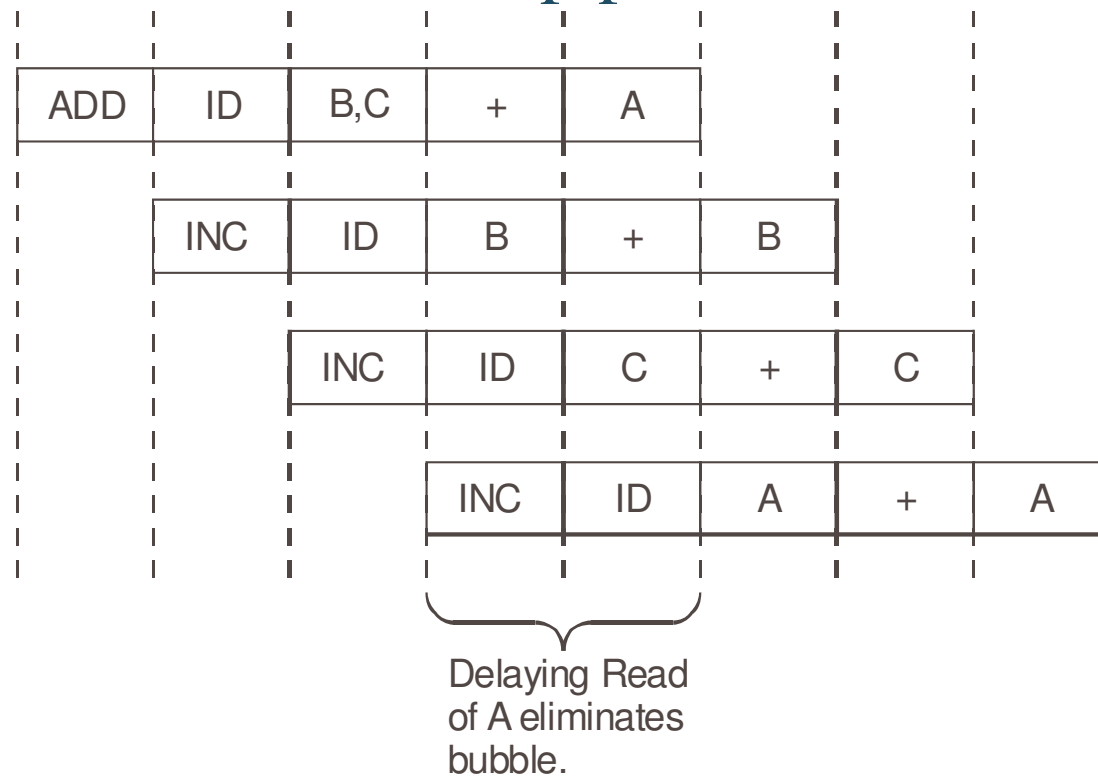
INC b

INC c

INC a

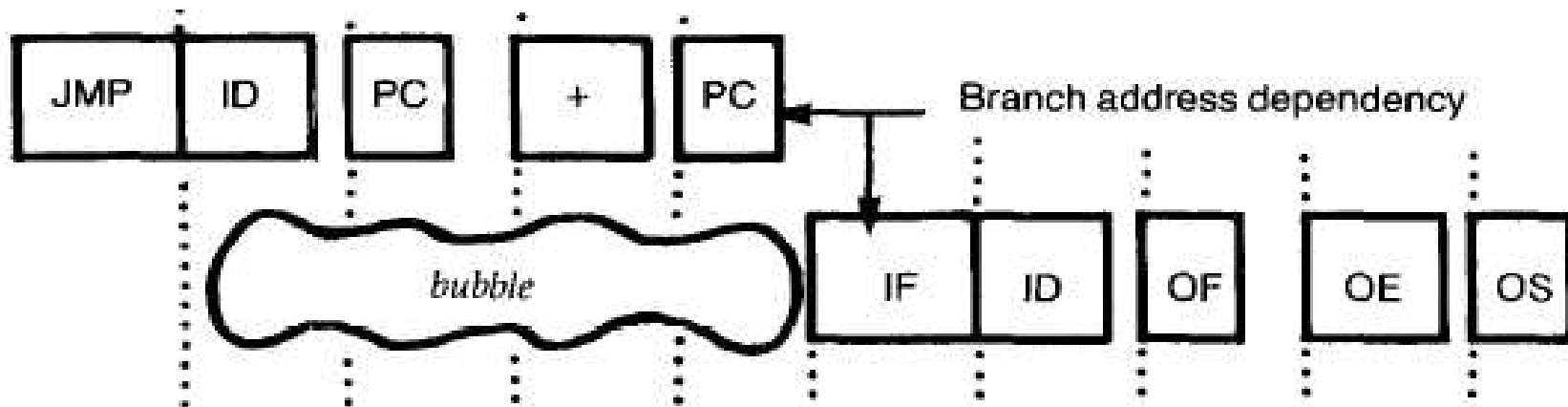
Rearranging Instructions

- ◆ By rearranging instructions, it is possible to eliminate at least some pipeline bubbles:



Pipelines and Branching

- ◆ Branch instructions cause a problem in a pipeline because the branch address may not be known until near the end of the pipe:





Chinks in the RISC Armor

- ◆ RISC is not a perfect technology.
 - The RISC v. CISC argument is part technology, part religion.
- ◆ Technology enhancements that help make RISC machines faster may also help CISC machines run faster.
- ◆ One classic argument is that the code size of a RISC machine will tend to be much larger than that of a CISC machine.
- ◆ Another argument is that compiler technology has advanced.

Program Size Comparisons

- ◆ It is difficult to fairly compare RISC and CISC machines:

Program	Source Lines	MAXVAX		MIDVAX		MINVAX	
		Size	Relative	Size	Relative	Size	Relative
Bq	40	1979	1.00	2271	1.14	2979	1.50
find	320	3625	1.00	4665	1.30	5801	1.61
ypp	898	10071	1.00	15931	1.58	25983	2.58
y	3599	37199	1.00	61219	1.65	100203	2.69
wf	169	3075	1.00	3779	1.22	5395	1.75
xd	1289	16349	1.00	23215	1.42	42671	2.61
sr	1403	14321	1.00	20336	1.42	31936	2.23
roff	2134	24025	1.00	40842	1.70	59103	2.46
spo	187	3246	1.00	4707	1.45	8147	2.51
od	723	9736	1.00	13631	1.40	25506	2.62
Average	1076	12363	1.00	19060	1.54	30773	2.48

Source: Davidson, "The effect of instruction set complexity on code size"

Cache Performance Comparison

- ◆ Differences between architectures may not be significant in some ways:

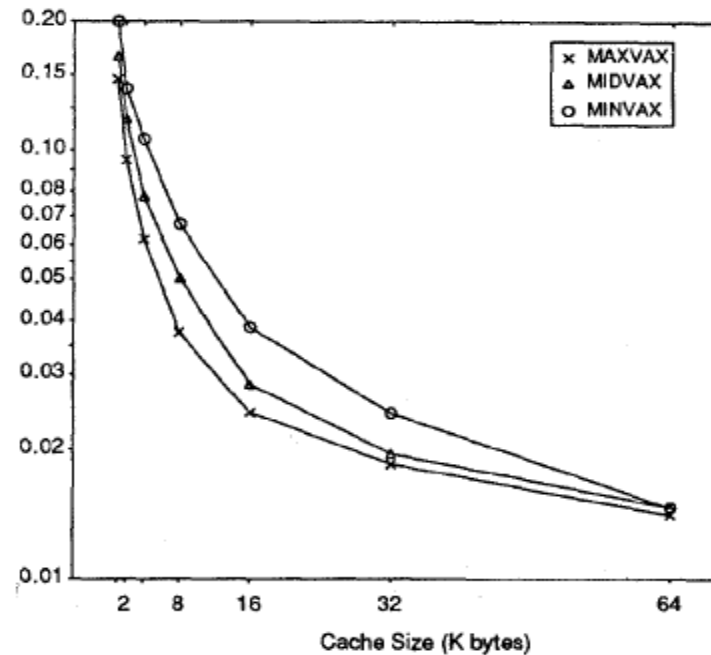


Figure 1. Miss Ratio vs. Cache Size for MAXVAX, MIDVAX, and MINVAX machines.

Source: Davidson, "The effect of instruction set complexity on code size"

- ◆ Memory activity is significantly different between architectures:

Cache Capacity	MAXVAX		MIDVAX		MINVAX	
	Mem. Reads	Relative	Mem. Reads	Relative	Mem. Reads	Relative
1k	174976	1.00	255860	1.48	471216	2.69
2k	116038	1.00	184106	1.58	331350	2.85
4k	73354	1.00	119340	1.62	253840	3.45
8k	42902	1.00	74848	1.74	159546	3.72
16k	25500	1.00	40198	1.57	88770	3.48
32k	17750	1.00	25574	1.44	51860	2.92
64k	13150	1.00	17024	1.29	28446	2.16
128k	12130	1.00	14556	1.20	24381	2.01
256k	11467	1.00	13416	1.17	21213	1.85

Table II. Bus Traffic and Ratio to MAXVAX.