CS 5803
Introduction to High Performance Computer Architecture: RISC vs. CISC

A.R. Hurson
323 Computer Science Building,
Missouri S&T
hurson@mst.edu

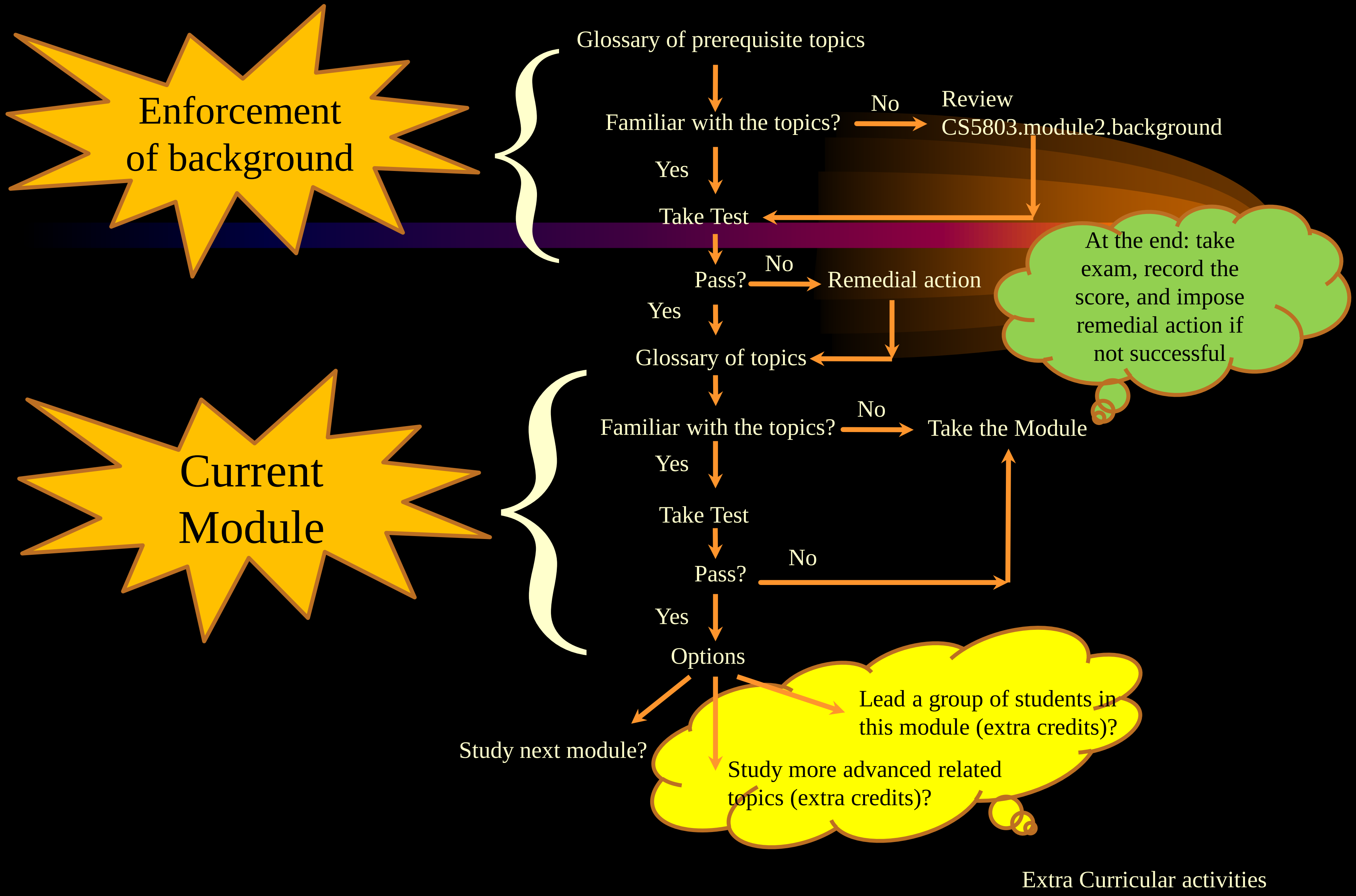
Introduction to High Performance Computer Architecture

- Outline
  - How to improve CPU time?
  - Complex Instruction Set Computer (CISC) - Historical Perspective
  - Complex Instruction Set Computer (CISC) - Characteristics
  - Reduced Instruction Set Computer (RISC) - Historical Perspective
  - Reduced Instruction Set Computer (CISC) – Characteristics
  - RISC *vs.* CISC

Note, this unit will be covered in three lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS5803.module3
2) Study the supplement module (supplement CS5803.module2)
3) Act as a helper to help other students in studying CS5803.module2

Note, options 2 and 3 have extra credits as noted in course outline.



Enforcement of background

Glossary of prerequisite topics

Familiar with the topics?

No

Review CS5803.module2.background

Yes

Take Test

Pass?

No

Remedial action

Yes

Glossary of topics

At the end: take exam, record the score, and impose remedial action if not successful

Current Module

Familiar with the topics?

No

Take the Module

Yes

Take Test

Pass?

No

Yes

Options

Study next module?

Study more advanced related topics (extra credits)?

Lead a group of students in this module (extra credits)?

Extra Curricular activities

- Question
  - With respect to our earlier definition of *CPU* time, discuss how the performance can be improved?

$$T = I_c * CPI * \tau = \sum_{i=1}^{n} \left(CPI_i * I_i\right) * \tau$$

- In response to this question, the *CPU* time can be reduced either by reducing the I_C and/or *CPI*.
- Note the performance improvement due to the advances in technology to reduce τ is beyond the scope of this discussion.

- Two Design philosophies
  - I_C can be reduced by increasing the functionality of the system — increasing the instruction set by allowing hardware support for more complex instructions.
  - This design pattern results in the so-called complex instruction set computer (CISC).

- Two Design philosophies
  - *CPI* can be reduced by allowing hardware support for just simple instructions.
  - This design pattern results in the so-called reduced instruction set computer (RISC).

- Two Design philosophies
  - In an effort to improve the performance one design philosophy suggest complexity and the other suggest simplicity!

Complex Instruction Set Computer

- The introduction of the IBM System/360 family was the beginning of modern computer technology — a series of computers with different levels of performance for different prices, all running identical software (Compatibility).

- As noted before, this originated the distinction between computer architecture and hardware.

- Complex Instruction Set Computer
  - Micro-programming was the primary technological innovation behind this new marketing concept — i.e., Computer Family.
  - Micro-programming relied on a small control memory and was an elegant way of building the processor control unit for a large instruction set.

Complex Instruction Set Computer

- The main memory of these systems were magnetic core memories.
- The small control memories were based on a technology about 10 times faster than core memory.
- The rapid growth of semiconductor memory also influenced the implementation of micro-programming at the mini and micro computer levels.

- Complex Instruction Set Computer
  - Due to the high cost and low performance of magnetic core memory, memory efficiency was the dominating concern in the previous metric parameters — execution speed was proportional to the program size.
  - This belief led to the invention of many instruction formats that reduced program size.

Complex Instruction Set Computer

- The rapid growth of integrated technology, along with the belief that execution time is proportional to the program size, motivated the following design principles:
  - Large control memory would add little or nothing to the cost of the machine.
  - Moving software functions to micro code would result in faster computer and more reliable functions.

Complex Instruction Set Computer

- Architectural techniques that led to smaller programs also led to faster computers.
- Stacks or memory-to-memory architectures were superior execution models.

- Complex Instruction Set Computer
  - Let us look at the translation of A ⇐ B + C in three execution models:

◆ Complex Instruction Set Computer

- Memory-to-Memory organization

8	16	16	16
ADD	B	C	A

Instruction Length	56 bits
Data Size	96 bits (data words are 32 bits each)
Total Memory	152 bits

Complex Instruction Set Computer

Register-to-Register Organization

8	4	16		
Load	r_B	B		
Load	r_C	C		
ADD	r_A	r_B	r_C	
Store	r_A	A		

Instruction Length	104 bits
Data Size	96 bits
Total Memory	200 bits

Complex Instruction Set Computer

- Memory-to-Register Organization

8	16
Load	B
ADD	C
Store	A

Instruction Length	72 bits
Data Size	96 bits
Total Memory	168 bits

Complex Instruction Set Computer

- In general, Complex Instruction Set Computer (CISC) supports:
  - Relatively large instruction set containing some complex and time consuming instructions.
  - Large number of addressing modes.
  - Large number of instruction formats.

- Complex Instruction Set Computer
  - VAX 11/780 Architectural Features
    - It is a 32-bit machine.
    - It has an instruction set of size 303.
    - It supports different data types:
      - Integer:byte, word, long word, Quad word, octa word.
      - Floating point: 32-bit-8-bit exponent, 64-bit-8-bit exponent, 64-bit-11-bit exponent, 128-bit-15-bit exponent.
      - Packed decimal.
      - Character String.
      - Variable length bit field.

- Complex Instruction Set Computer
  - VAX 11/780 Architectural Features
    - It supports 16 different addressing modes.
    - It supports several instruction formats:

      op.code, $\{operand_i\}$ $0 \le i \le 6$

Complex Instruction Set Computer

Characteristics of Some Computer Architectures

	IBM 370/168	VAX 11/780	Dorado	iAPX-432
Year	1973	1978	1978	1982
Number of Instructions	208	303	270	222
Control Memory Size	420 Kbits	480 Kbits	136 Kbits	64 Kbits
Instruction Size	16-48	16-456	8-24	6-321
Technology	ECL MSI	TTL MSI	ECL MSI	NMOS VLSI
Execution Model	Register-Memory, Memory-Memory, Register-Register	Register-Memory, Memory-Memory, Register-Register	Stack	Stack, Memory-Memory
Cache Size	64 Kbits	64 Kbits	64 Kbits	0

Complex Instruction Set Computer

- With the continuing growth of semiconductor memory, the architecture research community argued for richer instruction sets:
  - Richer instruction sets would simplify compilers,
  - Richer instruction sets would alleviate the software crisis,
  - Richer instruction sets would improve architectural quality.

- Complex Instruction Set Computer
  - To support a machine with a complex instruction set one is required to develop a very complex and sophisticated control unit to differentiate between the numerous options available in order to activate appropriate control signals.

Complex Instruction Set Computer

- To summarize:
  - Slow access to memory motivated an architectural design that reduced number of accesses to the main memory. This was achieved by supporting a large variety of instructions and addressing modes.
  - Complexity of the control unit and compatibility of various architectures were supported through micro-programming.

Complex Instruction Set Computer

- CISC Philosophy results in:
  - Complex Control Units:
    - Large design time,
    - Higher probability of errors,
    - Harder to locate and correct faults,
    - Longer instruction cycle, and
    - Longer clock cycle.
  - More complex compiler
  - Lower hardware utilization — creation of redundant features.

From CISC to RISC

- As discussed before, computer architects were reaching some design principles (CISC), however, the implementation world was changing:
  - Semiconductor memory was replacing core memory — As a result, main memories would no longer be 10 times slower than control memories.
  - Control store ROMs were changing to control store RAMs — Since it was practically impossible to develop a large error free micro-program.

From CISC to RISC

- Cache memories were included in the architectures.
- Compilers found it difficult to help close the semantic gap — Attempts to close the semantic gap had actually introduced a performance gap.

- As a result some computer architects got motivated to reevaluate the adapted architectural design principles.

- Summary
  - CISC and RISC
    - Motivation for CISC
    - CISC characteristics
    - Transition from CISC to RISC
  - RISC
    - RISC characteristics

Reduced Instruction Set Computer

- Functions should be kept simple unless there is a very good reason to do otherwise.
- Micro instructions should not be faster than simple instructions.
- 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.

- Reduced Instruction Set Computer
  - Functions should be kept simple
    - The effective speed of a computer can be maximized by migrating all but the most frequently used functions into software.
    - Included in hardware are only those performance features that are pointed to by dynamic studies of high level language programs.

- Reduced Instruction Set Computer
  - Functions should be kept simple
    - A resource is incorporated in the architecture only if its incorporation is justified by its frequency of use, and if its incorporation does not slow down other resources that are used more frequently.

Reduced Instruction Set Computer

- Simplicity of the instruction set and addressing modes results in a small, fast and relatively easily to design decoder to analyze the instructions. This allows one to effectively develop an instruction pipeline.
- This results in the execution of one instruction per pipeline pulse — *CPI* equal to 1.

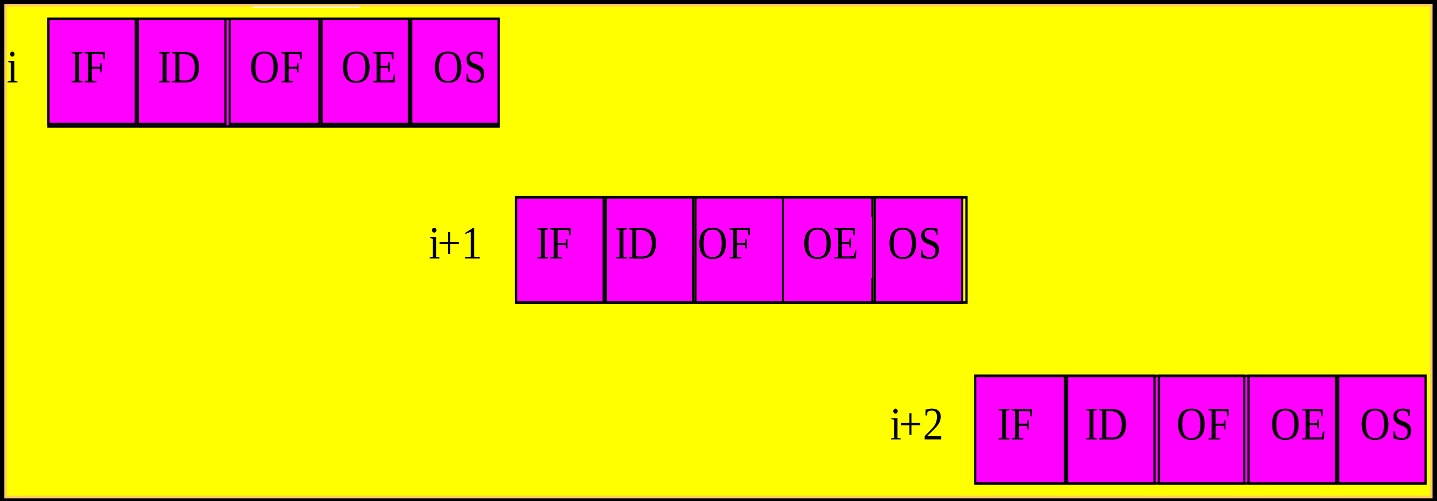
Reduced Instruction Set Computer

Instruction Pipelining

- Assume an instruction cycle can be partitioned into the following stages:

IF	: instruction fetch
OE	: operand execute
ID	: instruction decode
OS	: operand store
OF	: Operand fetch

- Reduced Instruction Set Computer
  - Non-pipelined instruction cycle:



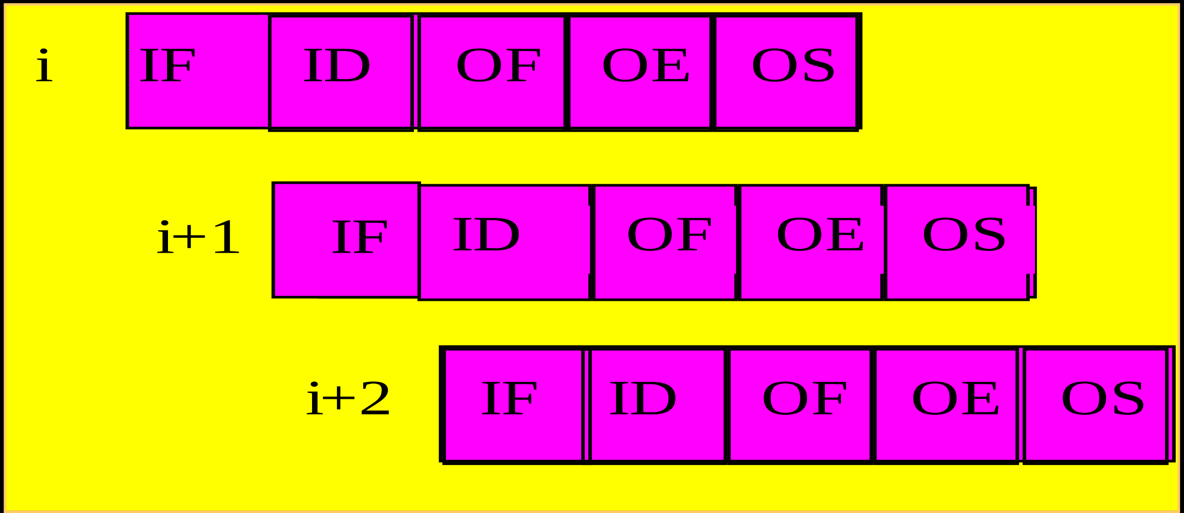
i IF ID OF OE OS

i+1 IF ID OF OE OS

i+2 IF ID OF OE OS

Reduced Instruction Set Computer

- Pipelined instruction cycle:



i IF ID OF OE OS

i+1 IF ID OF OE OS

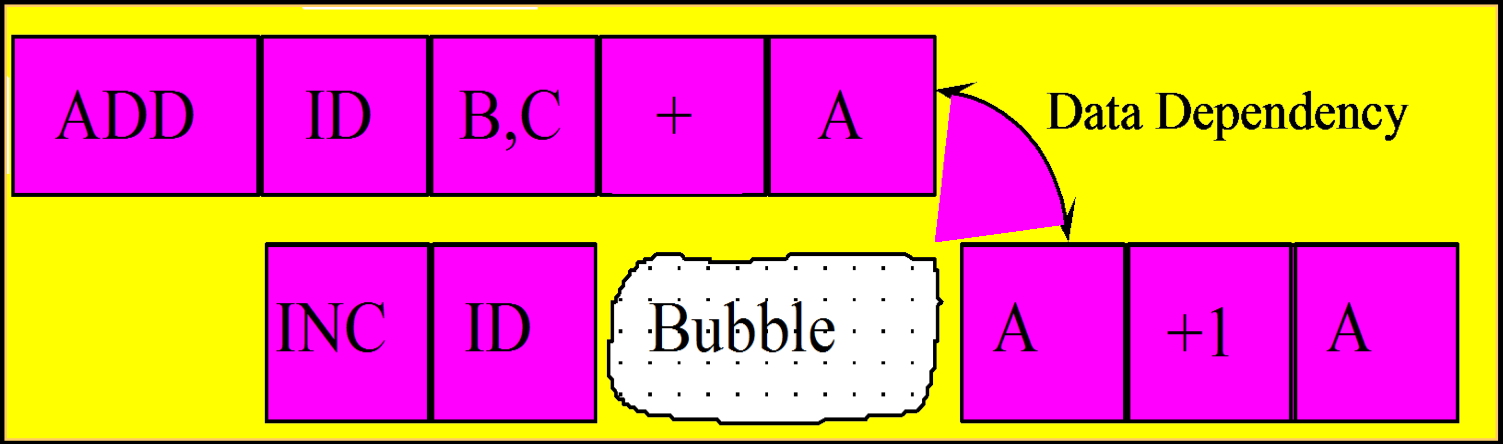
i+2 IF ID OF OE OS

Reduced Instruction Set Computer

- Pipelined instruction cycle:
  - Naturally, instruction pipelining is not without its own problems. A concept known as hazard effects the performance of a pipeline organization.
  - Related to our discussion, in the following two types of hazard will be discussed.

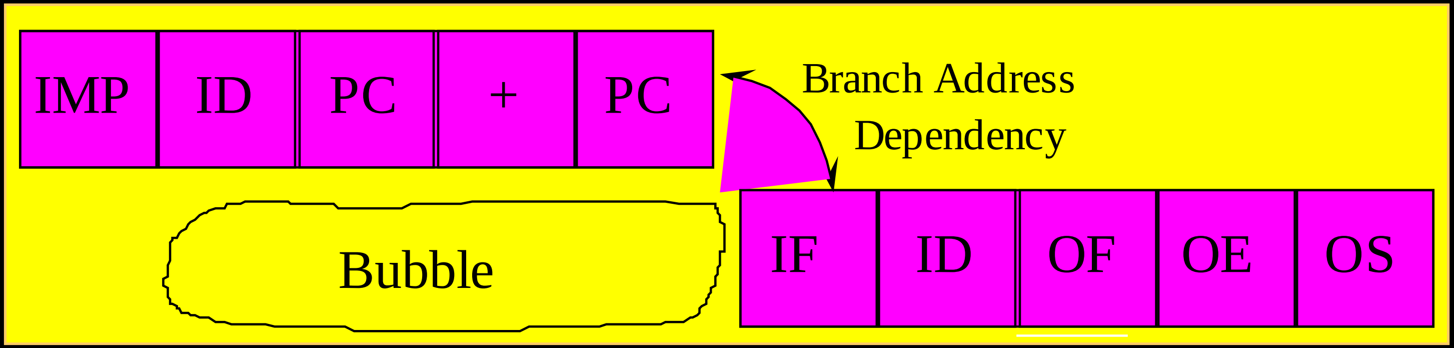
Reduced Instruction Set Computer

- Pipelined instruction cycle — Pipelined with data interlock (Data Dependence Hazard)



Reduced Instruction Set Computer

- Pipelined instruction cycle — Pipelined with branch interlock (Control Dependence Hazard)



Reduced Instruction Set Computer

- Delayed Branch — To get the advantage of the instruction pipeline, it would be necessary to insert a NO-OP operation — in the worst case every branch would take several NO-OP instructions.
- A better approach is to find some independent instructions and switch the order of the instructions in the program.

Reduced Instruction Set Computer

Address	Traditional Machine	Delayed Branch	Optimized Delayed Branch
100	Load X,A	Load X,A	Load X,A
101	ADD 1,A	ADD 1,A	JMP 105
102	JMP 105	JMP 106	ADD 1,A
103	ADD A,B	NO-OP	ADD A,B
104	SUB C,B	ADD A,B	SUB C,B
105	Store A,Z	SUB C,B	Store A,Z
106		Store A,Z	

- The branch instruction is not data dependent on the *ADD* at address 101, so switching the *JMP* and *ADD* results an equivalent result.

Reduced Instruction Set Computer

- Common RISC Features
  - Operations are register-to-register with only LOAD and STORE instructions to access memory.
  - The operations and addressing modes are reduced.
  - Instruction formats are simple and do not cross word boundaries.
  - RISC branches avoid pipeline penalties.

Reduced Instruction Set Computer

Architectural Features of Some Earlier RISC Machines

	IBM 801	**RISC1**	**MIPS**
Year	1980	1982	1983
Number of Instructions	120	39	55
Control Memory Size	0	0	0
Instruction Length	32	32	32
Technology	ECL MSI	NMOS, VLSI	NMOS, VLSI
Execution Model	Register-Register	Register-Register	Register-Register

CISC Characteristics

- Instruction set usually larger than 100,
- Number of addressing modes supported is usually larger than 4,
- Number of instruction formats supported in usually larger than 4,
- Most instructions require multiple cycles for execution,

- CISC Characteristics
  - Support of memory-to-memory model of execution,
  - Existence of special purpose registers,
  - Micro-programmed control unit, and
  - Machine instructions at a relatively high level, which is close to the level of high level language statements.

RISC Characteristics

- Most instructions require single cycle for execution,
- Memory is accessed just through LOAD and STORE instructions,
- Hardwired control unit,
- Supports relatively few instruction formats and addressing modes,

RISC Characteristics

- Fixed instruction length format,
- Highly pipelined instruction cycle,
- Large number of on chip registers,
- Instruction set is targeted for a specific application, and
- Use of co-processor for complex operations requiring hardware support.

- RISC vs. CISC
  - RISC allows performance improvement through careful design of the data path, pipeline and other CPU resources.
  - RISC offers fewer clock cycles per instruction.
  - RISC is more suitable for current technology.
  - RISC might imply specialization.

- RISC vs. CISC
  - For the same program RISC requires larger assembly code than CISC.
  - From software reliability point of view, RISC has a disadvantage.

- RISC vs. CISC
  - Back to our original question — based on the equation

$$T = I_c * CPI * \tau = I_c * (p+m*k)* \tau$$

what is a better design philosophy, RISC or CISC?

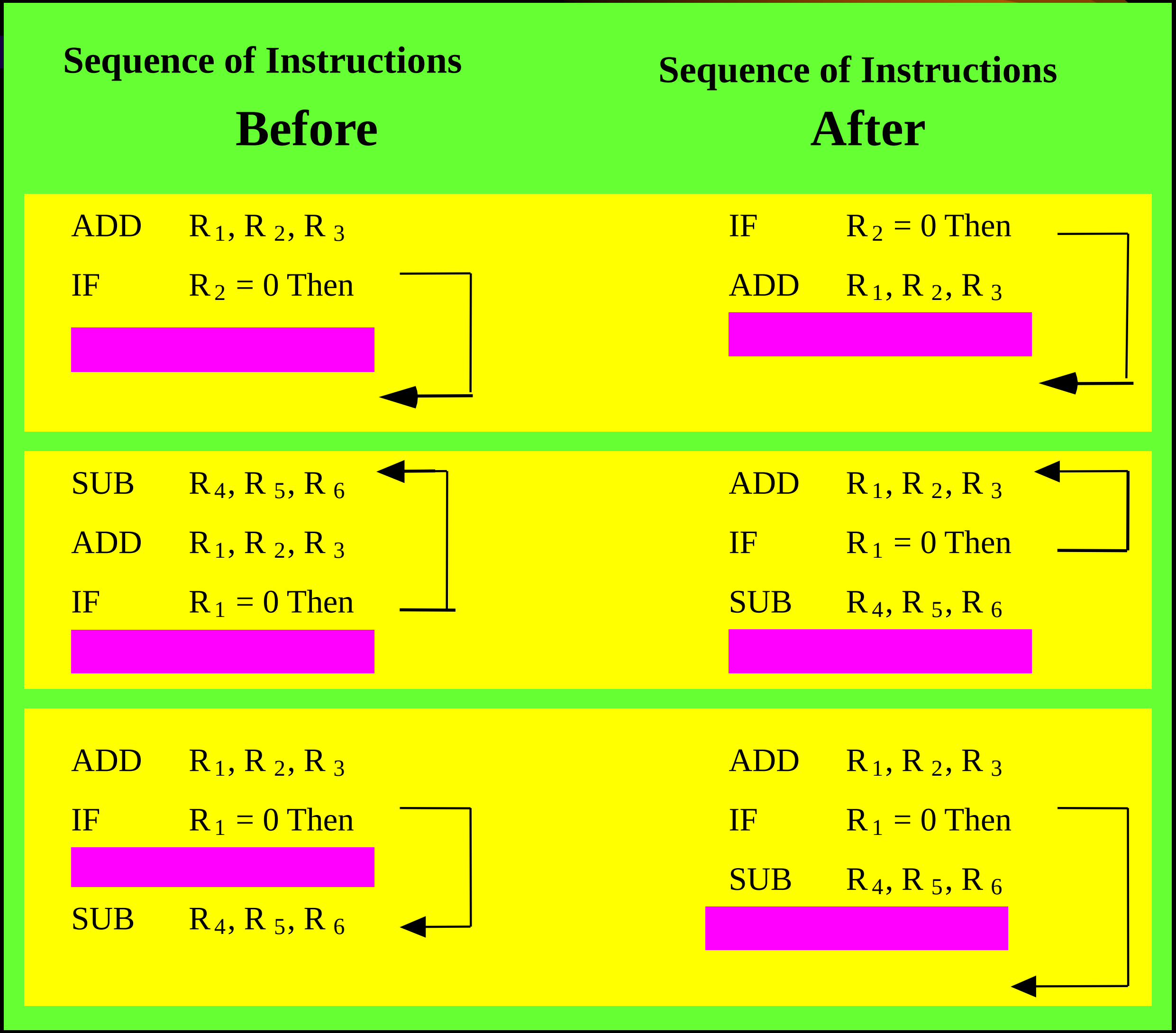
Reduced Instruction Set Computer

- As noted before, a RISC concept in the best case would allow a CPI equal to 1. Is it possible to break this barrier?
- Reducing the clock cycle time and hence increasing the frequency is one way to improve the performance. Address the shortcoming(s) of this approach.

- Questions
  - True or False: shorter length instructions imply faster processor (why)?
  - Length of the operation code affects the length of the instructions. Define two schemes which allows one to reduce the length of the op-code.
  - Name and explain different factors which affect the length and format of an instruction.

- Problem
  - Within the scope of RISC, use delay branch technique to reduce pipeline penalties (instruction format — op-code, destination, source1, source2): Justify your answer.
  - Assuming new value of PC is determined in ID stage and instructions in the block are independent of R_4, R_5, and R_6, then we have:

- Problem



Sequence of Instructions

Before

Sequence of Instructions

After

ADD R_1, R_2, R_3
IF $R_2 = 0$ Then

IF $R_2 = 0$ Then
ADD R_1, R_2, R_3

SUB R_4, R_5, R_6
ADD R_1, R_2, R_3
IF $R_1 = 0$ Then

ADD R_1, R_2, R_3
IF $R_1 = 0$ Then
SUB R_4, R_5, R_6

ADD R_1, R_2, R_3
IF $R_1 = 0$ Then
SUB R_4, R_5, R_6

ADD R_1, R_2, R_3
IF $R_1 = 0$ Then
SUB R_4, R_5, R_6