Computer Organization

Performance,
RISC and CISC

Department of Computer Science
Missouri University of Science & Technology
hurson@mst.edu

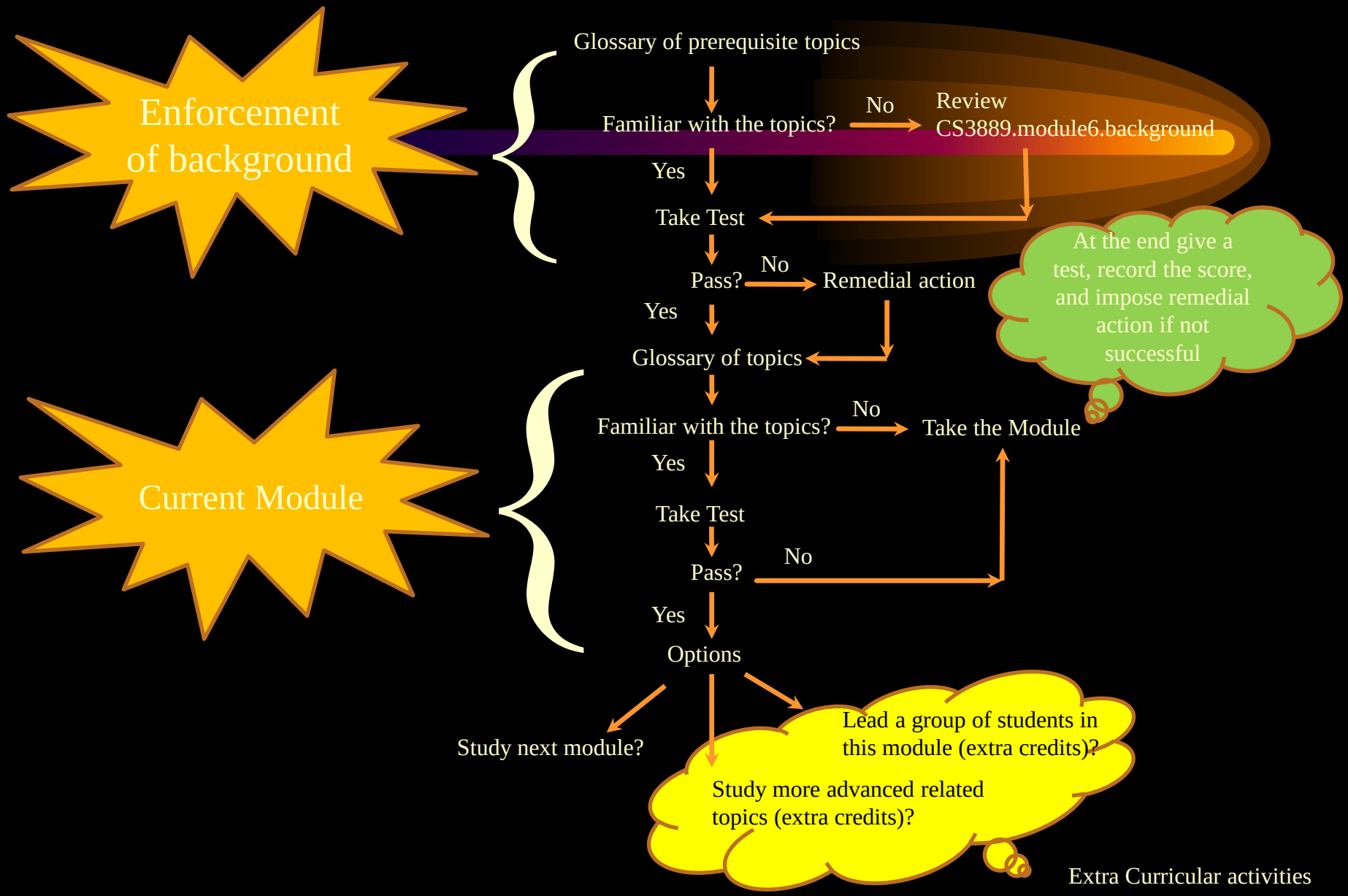
Computer Organization

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 CS3889.module7
2) Study the supplement module (supplement CS3889.module6)
3) Act as a helper to help other students in studying CS3889.module6

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

Computer Organization



Computer Organization

- Performance Metrics
  - Response Time (Execution time, Latency) — The time elapse between the start and the completion of an event.
  - Throughput (Bandwidth) — The amount of work done in a given time.
  - Performance — Number of events occurring per unit of time.

Computer Organization

- Performance Metrics
  - Note execution time is the reciprocal of performance — lower execution time implies higher performance.
  - Note Response time, Throughput, and Performance are all closely related to each other.

- Performance Metrics
  - A system (*X*) is faster than (*Y*), if for a given task, the response time on *X* is lower than on *Y*.

$$n = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = \frac{\frac{1}{\text{Performance}_Y}}{\frac{1}{\text{Performance}_X}} = \frac{\text{Performance}_X}{\text{Performance}_Y}$$

- Performance Metrics — Example
  - Machine *A* runs a program in 10 seconds and machine *B* runs the same program in 15 seconds. Therefore:

$$n = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \left(\frac{15}{10}\right) = 1.5$$

Computer Organization

- Performance Metrics
  - Average Execution time — Equal probability of running programs in the workload

$$\frac{1}{n}\sum_{i=1}^{n} Time_i$$

Where $Time_i$ is the execution time of the i^{th} program
And n is the number of the program in the workload.

- Performance Metrics
  - Weighted Execution time — unequal mix of programs in the workload

    where $weight_i$ is the frequency of the i^{th} gram in the workload.

$$\sum_{i=1}^{n} Weight_i \times Time_i$$

Note $$\sum_{i=1}^{n} Weight_i = 1$$

- Performance Metrics
  - Million Instructions Per Second — MIPS is another performance measure to be used to evaluate computers.

$$MIPS = \frac{I_C}{\text{Execution time} * 10^6}$$

- Performance Metrics
  - Million Floating Point Operations Per Second — MFLOPS

    is another performance measure to be used to evaluate computers.

$$\text{MFLOPS} = \frac{\text{Number of floating point operations in a program}}{\text{Execution time} * 10^6}$$

- Performance Metrics
  - Response Time (Elapse time) — The latency to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, ...

- Performance Metrics
  - CPU time — The time the *CPU* is computing. It is further divided into:
    - User CPU time — The CPU time spent in the program,
    - System CPU time — The CPU time spent in operating system performing tasks requested by the program.

- Performance Metrics
  - The processor of today's computer is driven by a clock with a constant cycle time (τ).
  - The inverse of the cycle time is the clock rate (f).
  - The size of a program is determined by its instruction count (I_c) — number of the machine instructions to be executed.

- Performance Metrics
  - Let us define the average number of clock cycle per instruction (*CPI*) as:

$$CPI = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}} = \frac{\sum_{i=1}^{n} (CPI_i * I_i)}{I_c}$$

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

Where I_i is the number of time instruction i is executed and CPI_i is the average number of clock cycles for instruction i.

Computer Organization

- Performance Metrics
  - For a given instruction set, one can calculate the CPI over all instruction types, if the frequencies of the appearance of the instructions in the program is known.
  - *CPI* depends on the organization/architecture and the instruction set of the machine.
  - Clock rate depends on the technology and organization/architecture of the machine.
  - Instruction count depends on the instruction set of the machine and compiler technology.

Computer Organization

- Performance Metrics

$$\text{CPU}_{\text{Time}} = \text{CPU}_{\text{Clock cycles}} * \text{Clock}_{\text{Cycle time}}$$

$$\text{CPU}_{\text{Time}} = \frac{\text{CPU}_{\text{Clock cycles}}}{\text{Clock}_{\text{Rate}}}$$

The CPU time is estimated as

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

- Performance Metrics — Example
  - It takes 10 seconds to run a program on machine *A* that has a 400 MHz clock rate.
  - We are intended to build a faster machine that will run this program in 6 seconds. However, machine *B* requires 1.2 times as many clock cycles as machine *A* for this program. Calculate the clock rate of machine *B*:

- Performance Metrics — Example

$$CpuTime_A = CPUClockCycle_A / ClockRate_A$$

$$10 = CPUClockCycle_A / 400 * 10^6 \, Cycles/Second$$

$$CPUClockCycle_A = 4000 * 10^6$$

$$CPUTime_B = 1.2 * CPUClockCycle_A / ClockRate_B$$

$$ClockRate_B = 1.2 * 4000 * 10^6 / 6 = 800MHz$$

Performance Metrics — Example

- Suppose we have two implementation of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program.
- Which computer is faster for this program?

Performance Metrics — Example

- We know that both machines execute the same number of instructions, say I.

$$\text{CPU Time}_A = \text{CPUClockCycles}_A * \text{ClockCycleTime}_A = I * 2.0 * 250 \text{ ps} = 500 * I \text{ ps}$$

$$\text{CPU Time}_B = \text{CPUClockCycles}_B * \text{ClockCycleTime}_B = I * 1.2 * 500 \text{ ps} = 600 * I \text{ ps}$$

- Clearly A is faster than B.

Computer Organization

Performance Metrics — Example

- Assume we have the following information for different classes of operations:

	Class A	Class B	Class C
CPI	1	2	3

- Also assume the same code segment with different instruction mix:

Code Sequence	Instruction Counts		
	A	B	C
1	2	1	2
2	4	1	1

Performance Metrics — Example

- Refer to the previous example, which code is faster?
- $CPUClockCycle_1 = (2*1) + (1*2) + (2*3) = 10$
- $CPUClockCycle_2 = (4*1) + (1*2) + (1*3) = 9$
- So code sequence 2 is faster, however, note that code $sequence_2$ has more instructions than code $sequence_1$.

Computer Organization

- Performance Metrics
  - The execution of an instruction requires going through the instruction cycle. This involves the instruction fetch, decode, operand(s) fetch, execution, and store result(s):

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

Computer Organization

- Performance Metrics
  - *P* is the number of processor cycles needed to decode and execute the instruction, *m* is the number of the memory references needed, and *k* is the ratio between memory cycle time and processor cycle time, memory latency.

Computer Organization

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

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

- 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.

Computer Organization

- 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).

Computer Organization

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

Computer Organization

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

Computer Organization

- 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.

Computer Organization

- 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.

Computer Organization

- 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.

Computer Organization

- 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,

Computer Organization

- 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.

Computer Organization

- 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,

Computer Organization

- 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.

Computer Organization

- 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.

Computer Organization

- 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?

Computer Organization

- 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.