Performance

Dr. Prasenjit Chanak

Department of Computer Science and Engineering
Indian Institute of Technology (BHU), Varanasi
UP, 221005

Instruction Performance

- The performance equations above did not include any reference to the number of instructions needed for the program
- The compiler clearly generated instructions to execute, and the computer had to execute the instructions to run the program, the execution time must depend on the number of instructions in a program
- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

Instruction Performance

- Therefore, the number of clock cycles required for a program can be written as

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \begin{array}{c}\text{Average clock cycles} \\ \text{per instruction}\end{array}$$

- **Clock cycles per instruction (CPI)** Average number of clock cycles per instruction for a program or program fragment

The Classic CPU Performance Equation

- We can now write this basic performance equation in terms of **instruction count** (the number of instructions executed by the program), CPI, and clock cycle time:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

- or, since the clock rate is the inverse of clock cycle time:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

Pitfalls

- *Pitfall: Expecting the improvement of one aspect of a computer to increase overall performance by an amount proportional to the size of the improvement*
- The execution time of the program after making the improvement is given by the following simple equation known as **Amdahl's Law**:

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

- **Amdahl's Law:** A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the law of diminishing returns.

MIPS (million instructions per second)

- One alternative to time is **MIPS (million instructions per second)**. For a given program, MIPS is simply

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

- **Million instructions per second (MIPS):** A measurement of program execution speed based on the number of millions of instructions. MIPS is computed as the instruction count divided by the product of the execution time and 10^6.
- MIPS is an instruction execution rate, MIPS specifies performance inversely to execution time; faster computers have a higher MIPS rating.
- The good news about MIPS is that it is easy to understand, and faster computers mean bigger MIPS, which matches intuition

Contd.

- There are three problems with using MIPS as a measure for comparing computers.
- First, MIPS specifies the instruction execution rate but does not take into account the capabilities of the instructions. We cannot compare computers with different instruction sets using MIPS, since the instruction counts will certainly differ
- Second, MIPS varies between programs on the same computer; thus, a computer cannot have a single MIPS rating
- For example, by substituting for execution time, the relationship between MIPS, clock rate, and CPI:

$$\text{MIPS} = \frac{\text{Instruction count}}{\dfrac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

Instructions

Instruction

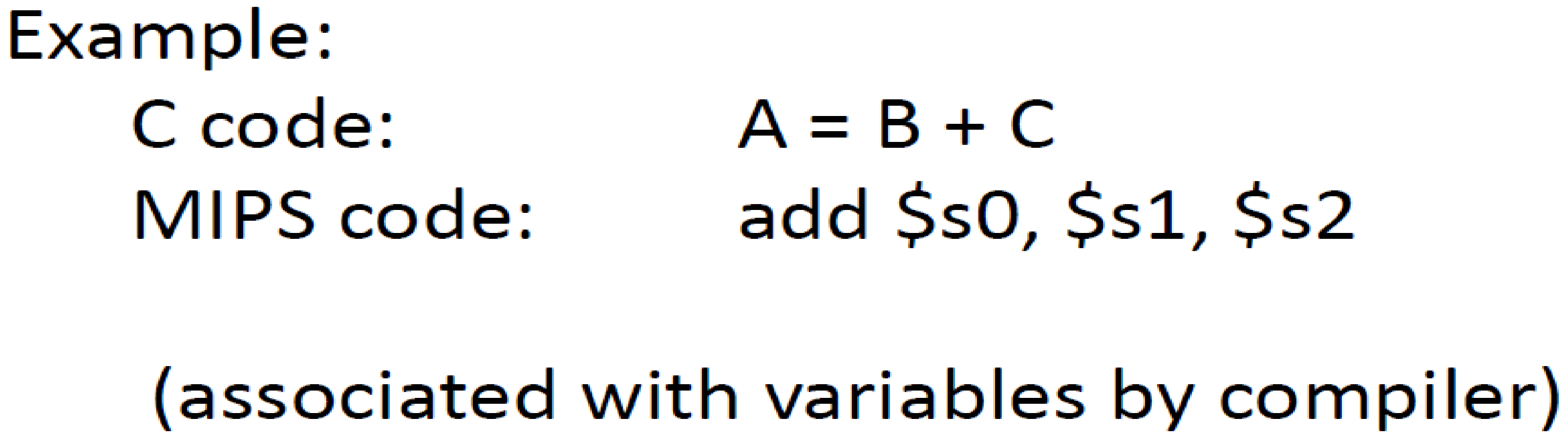
- To command a computer’s hardware, you must speak its language.
- The words of a computer’s language are called *instructions*, and its vocabulary is called an **instruction set**
- **MIPS** instruction sets designed since the 1980s
- Three other popular instruction sets:
  - ARMv7 is similar to MIPS. More than 9 billion chips with ARM processors were manufactured in 2011, making it the most popular instruction set in the world
  - The second example is the Intel x86, which powers both the PC and the cloud of the PostPC Era.
  - The third example is ARMv8, which extends the address size of the ARMv7 from 32 bits to 64 bits. Ironically, as we shall see, this 2013 instruction set is closer to MIPS than it is to ARMv7

Type of Instructions

- Instructions for **arithmetic**
- Instructions to **move data**
- Instructions for **decision making**
- Handling **constant operands**

MIPS Arithmetic

- All instructions have 3 operands
- Operand order is fixed (destination first)



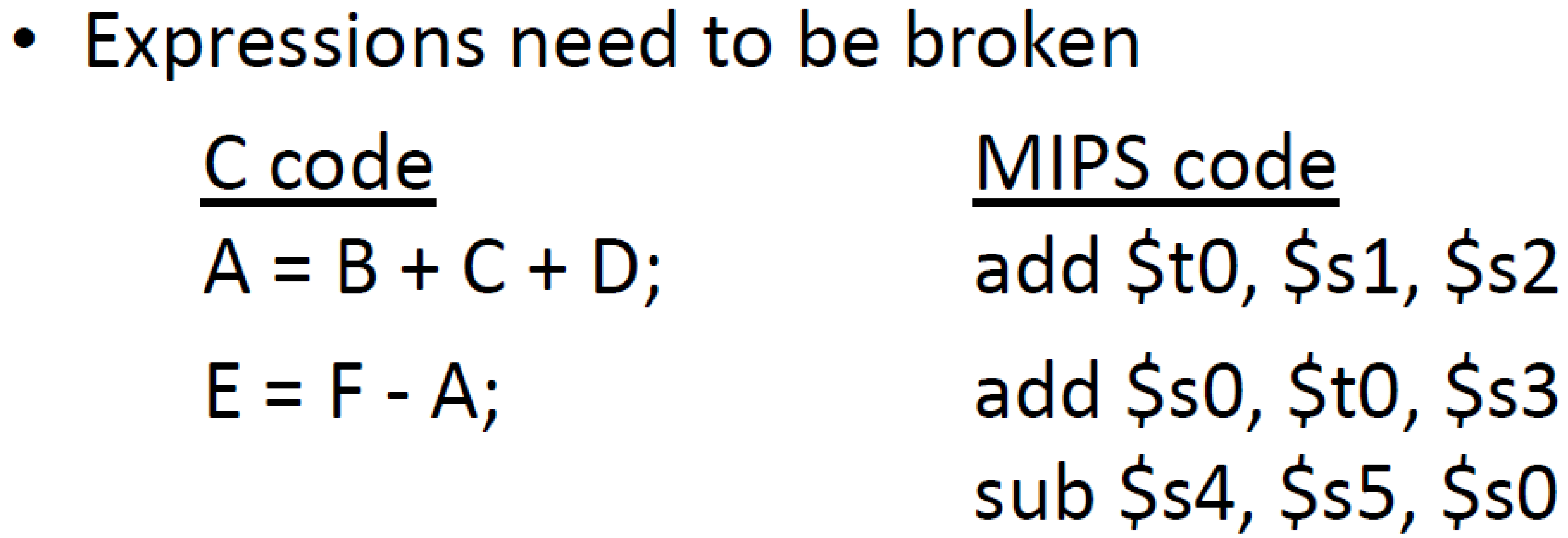
Example:

```
    C code:          A = B + C
    MIPS code:       add $s0, $s1, $s2
```

(associated with variables by compiler)

MIPS Arithmetic

- Simplicity favors regularity
- Operands must be registers, only 32 registers
- provided (smaller is faster)



- Expressions need to be broken

C code	MIPS code
A = B + C + D;	add $t0, $s1, $s2
E = F - A;	add $s0, $t0, $s3
	sub $s4, $s5, $s0

Register Names and Purpose

Name	Register number	Usage
`$zero`	0	the constant value 0
`$v0-$v1`	2-3	values for results
`$a0-$a3`	4-7	arguments
`$t0-$t7`	8-15	temporaries
`$s0-$s7`	16-23	saved
`$t8-$t9`	24-25	more temporaries
`$gp`	28	global pointer
`$sp`	29	stack pointer
`$fp`	30	frame pointer
`$ra`	31	return address