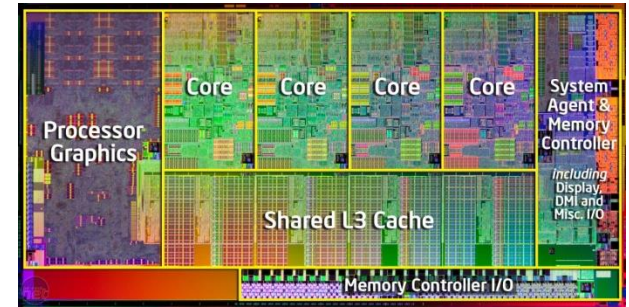




고려대학교
KOREA UNIVERSITY



COSE222, COMP212, CYDF210 Computer Architecture

Lecture 3. Performance

Prof. Taeweon Suh

Computer Science & Engineering

Korea University

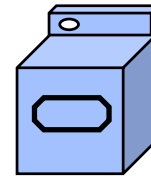
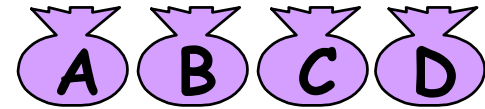
Response Time and Throughput

- How to measure performance of a computer?
 - **Response time (Execution time, Latency)**
 - Time between the start and the completion of a task
 - Important to individual users
 - **Embedded computers** and **PCs** are more focused on response time
 - **Throughput**
 - Total number of tasks done in a given time
 - Important to datacenter and/or supercomputer managers
 - **Servers** are more focused on throughput
- Need different performance metrics depending on machine types and/or usages

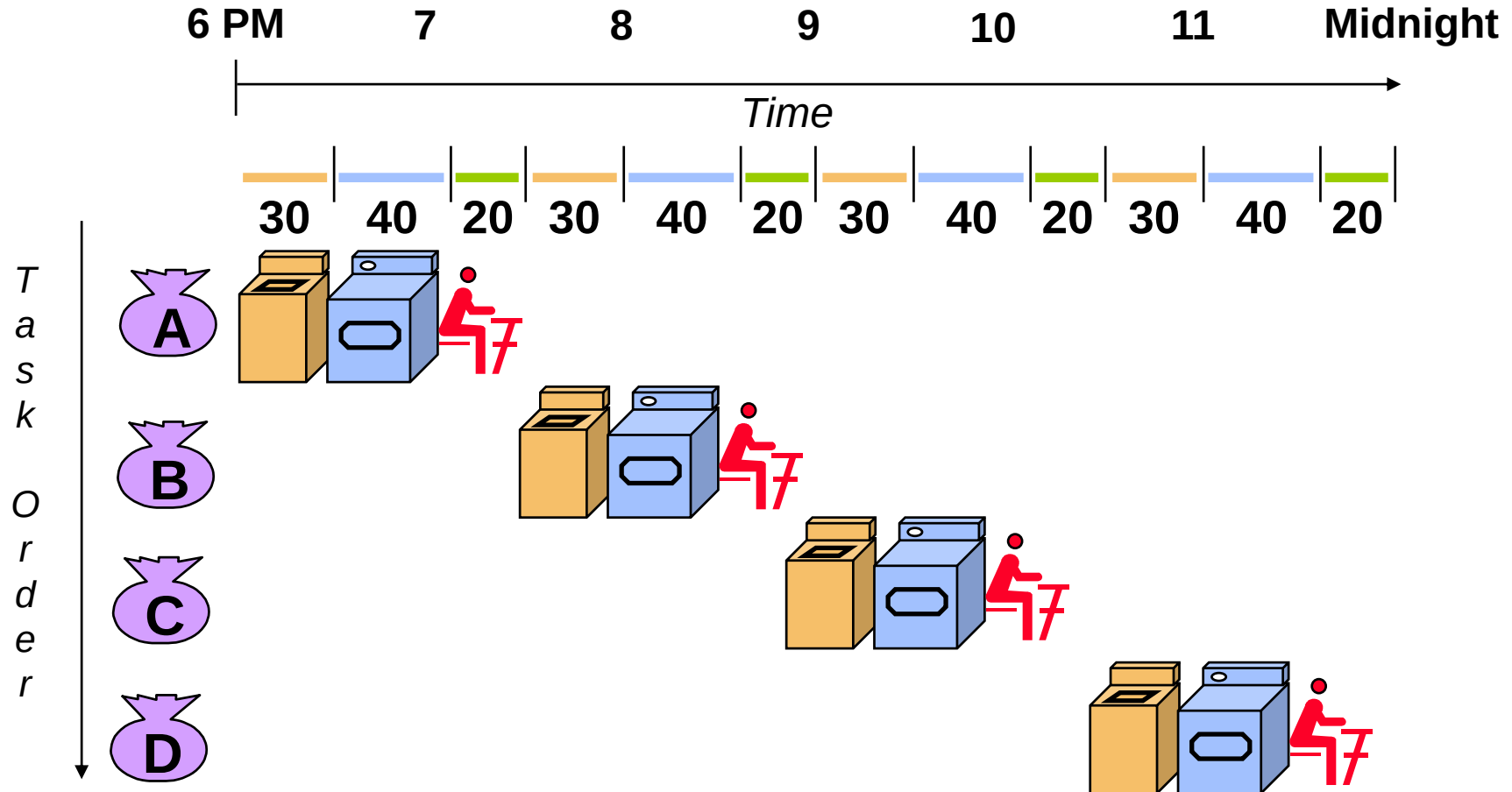
Response Time and Throughput

- Laundry Example

- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- **Washer** takes 30 minutes
- **Dryer** takes 40 minutes
- **Folder** takes 20 minutes

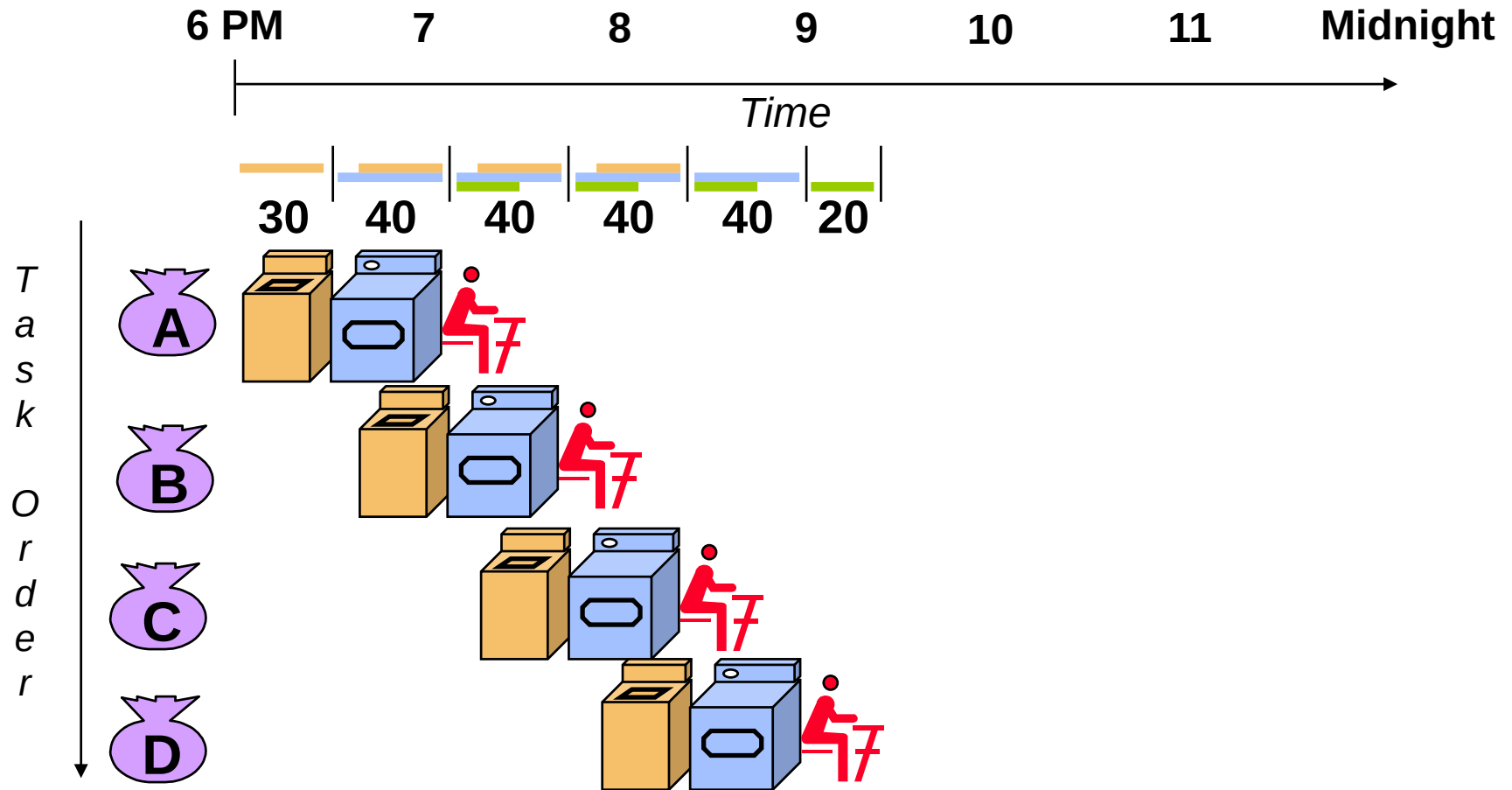


Sequential Laundry



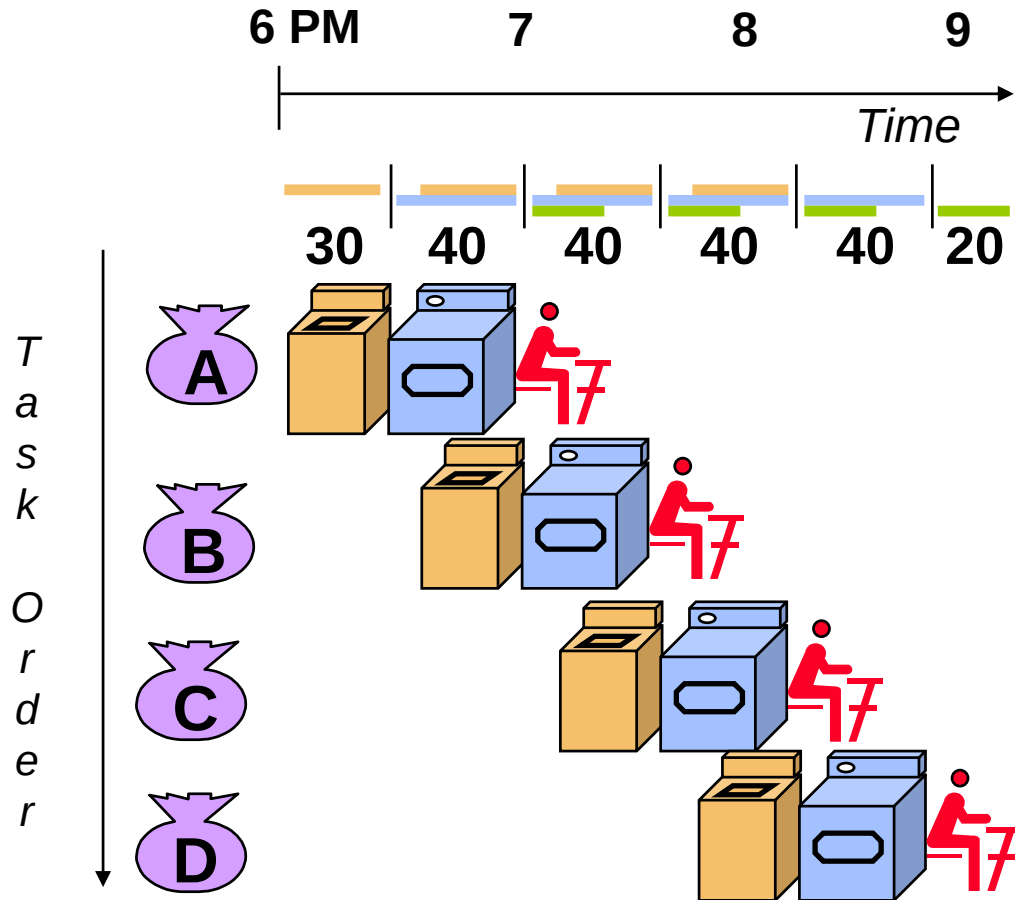
- Response time: 90 mins
- Throughput: 0.67 tasks / hr (= 90mins/task, 6 hours for 4 loads)

Pipelined Laundry



- Response time: 90 mins
- Throughput: 1.14 tasks / hr (= 52.5 mins/task, 3.5 hours for 4 loads)

Pipelining Lessons



- Pipelining **doesn't help latency (response time)** of a single task
- Pipelining **helps throughput** of entire workload
- Multiple tasks operating simultaneously
- Unbalanced lengths of pipeline stages reduce speedup
- Potential speedup = # of pipeline stages
- We are going to talk in detail about pipelining in chapter 4
 - The term project is to implement CPU with pipelining

-
- Let's focus on **response time** for now...

Relative Performance

- To maximize performance of your computer, you want to minimize execution time (response time) for a task
- Thus, we can relate performance and execution time for a computer X

$$\text{performance}_X = \frac{1}{\text{execution_time}_X}$$

If a computer X is n times faster than a computer Y ,

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution_time}_Y}{\text{execution_time}_X} = n$$

Example

- A computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds. How much is A faster than B?

$$\frac{\text{performance}_x}{\text{performance}_y} = \frac{\text{execution_time}_y}{\text{execution_time}_x} = n$$

The performance ratio is $\frac{15}{10} = 1.5$

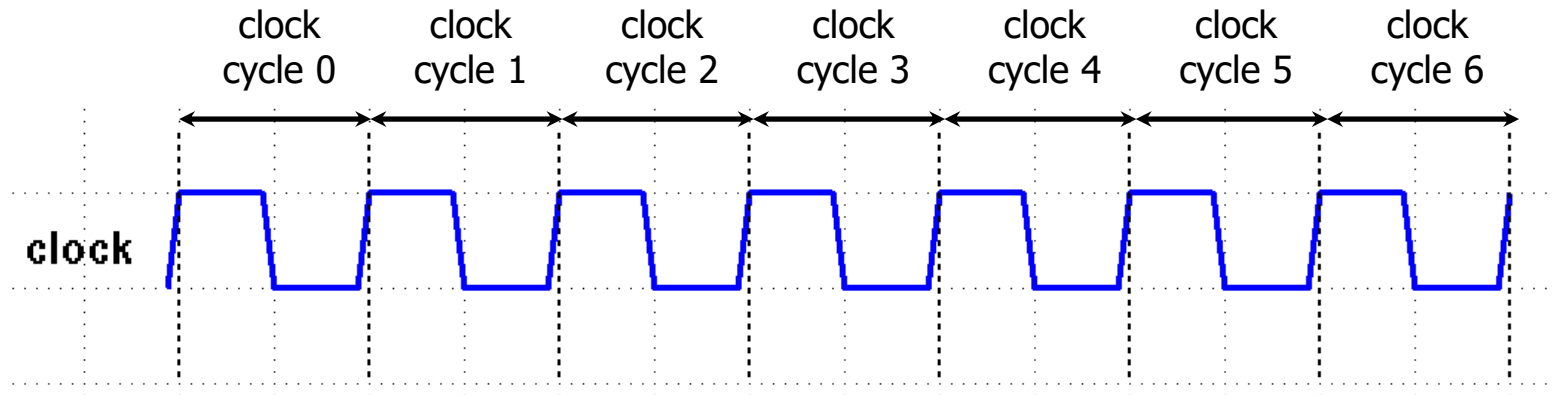
So, A is 1.5 times faster than B

Measuring Execution Time

- **Execution time** (elapsed time or wall-clock time) is measured in **seconds per program**
 - Total execution time includes all aspects: disk access, memory access, I/O activities, OS overhead
 - It determines the system performance
- **CPU time**
 - The time CPU spent processing a given job
 - It does not include time spent waiting for I/O, or running other programs

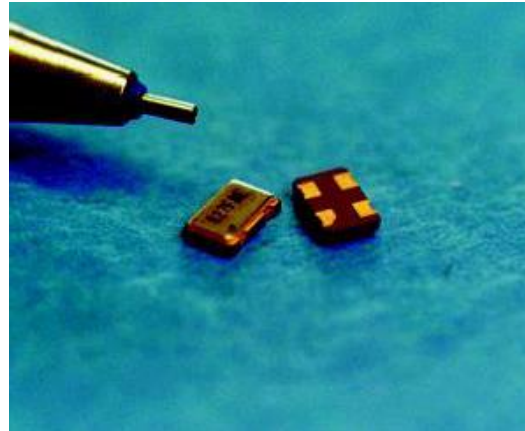
CPU Clock

- Let's use the CPU time for simplicity to measure performance
- Virtually all computers are constructed in sync with a clock
 - Discrete time intervals are called **clock cycles**



- **Clock period (T)**: duration of a clock cycle
 - e.g. $500\text{ps} = 0.5\text{ns} = 500 \times 10^{-12}\text{s}$
- **Clock frequency (f)**: clock cycles per second ($1/T$)
 - e.g. $1/T = 1/0.5\text{ns} = 2.0\text{GHz} = 2.0 \times 10^9\text{Hz}$

Reminder: Clock Oscillators



32.768kHz Crystals & Oscillators

Surface Mount Ultra Miniature packages for YOUR application

Offered in a variety of package and tolerance options, Abracon's 32.768kHz Precision Crystals & Oscillators are an essential part of virtually all timing applications. Abracon's impressive track record of high reliability products for Automotive Electronics uniquely positions itself to offer low-cost, high quality devices for most Industrial and high volume consumer electronics applications.

FEATURES:

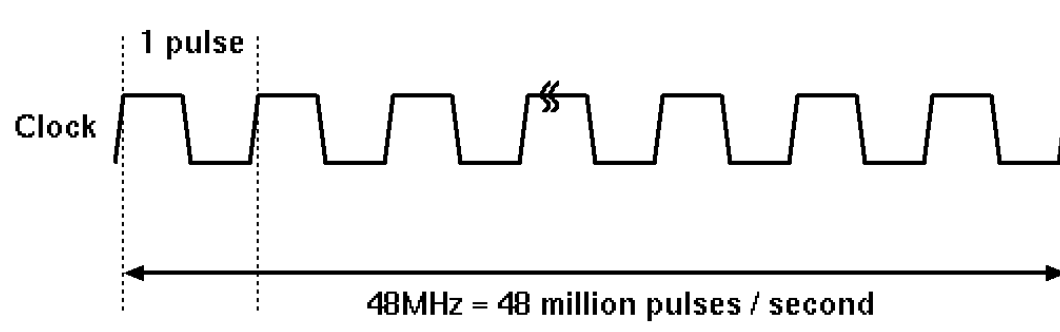
- 32.768kHz reference frequency
- Ultra miniature SMD packages
- Available $\pm 20.0\text{ppm}$ & $\pm 10.0\text{ppm}$ set tolerance
- Operating temperature range of -40°C $+85^{\circ}\text{C}$
- 12.5pf, 9pf, 7pf and 6pf standard crystal plating loads
- Oscillators available 20ppm -40°C $+85^{\circ}\text{C}$ with current consumption as low as 0.8 uA max

Authorized Stocking Distributors:

Visit www.abracon.com for additional listing of authorized distributors

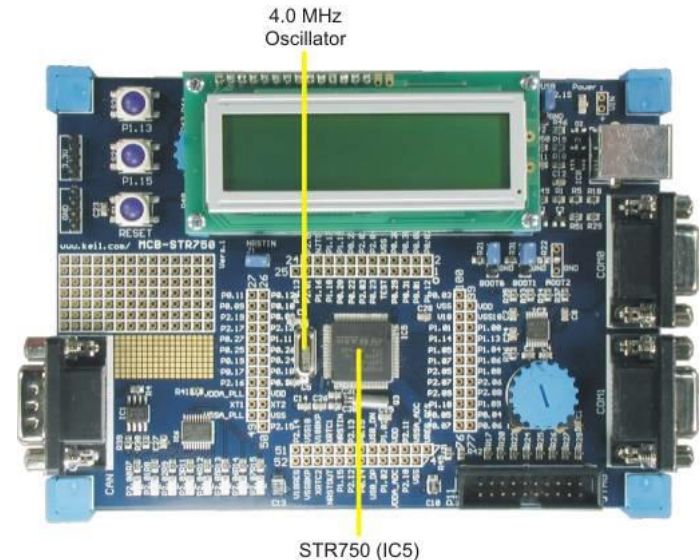
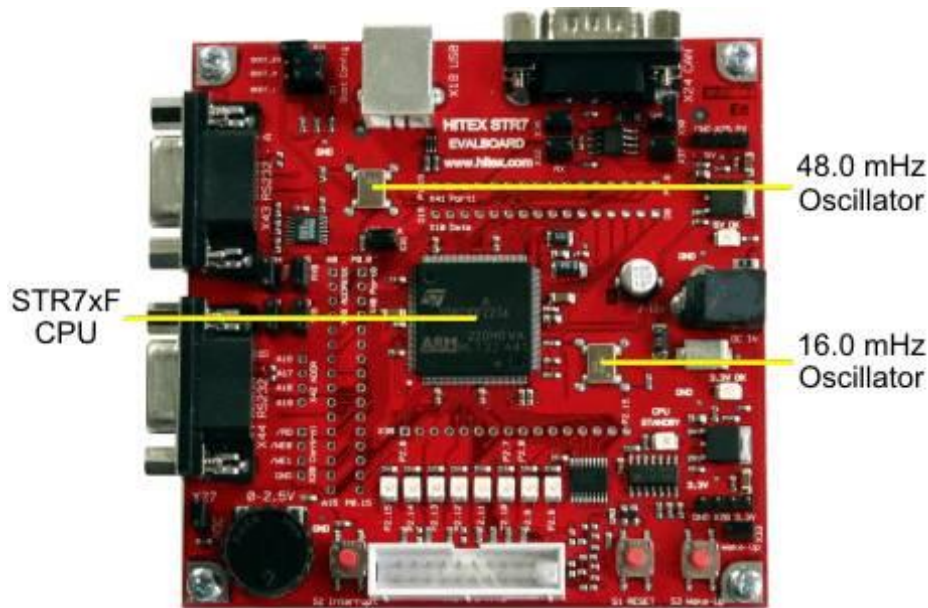
ABRACON CORPORATION
The Power of Linking Together

Product dimensions and part numbers are listed on the right side of the advertisement.

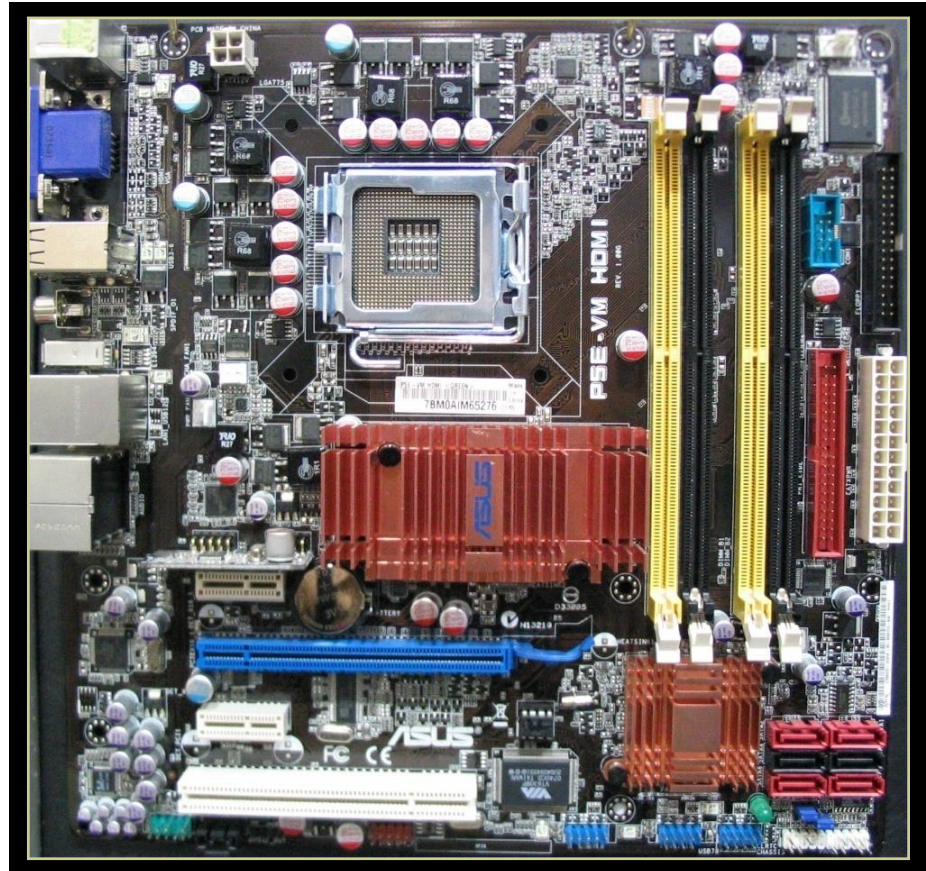


Reminder: Clock Oscillators in Digital Systems

- Virtually all digital systems are essentially synchronous to the clock



Where are clock oscillators?



CPU Time

- Express CPU time in terms of clock

$$\begin{aligned}\text{CPU Time} &= \text{CPU clock cycles} \times \text{clock cycle time (T)} \\ &= \frac{\text{CPU clock cycles}}{\text{Clock frequency (f)}}\end{aligned}$$

- So, the performance is improved by
 - Reducing the number of clock cycles
 - Increasing clock frequency

Example

- Computer A running at 2GHz requires 10 second CPU time to run your program
- Let's design a new Computer B
 - Aim for 6 second CPU time to run the same program
 - but causes $1.2 \times$ clock cycles, compared to Computer A
 - How fast should the computer B's clock (frequency) be?
 - Computer B requires 6 seconds to run the program
 $6 \text{ seconds} = (1.2 \times \text{CPU clock cycle}_A) / f$
 - How many clock cycles computer A needs?
 $10 \text{ sec} = \text{CPU clock cycle}_A / 2\text{GHz}$
 $\text{CPU clock cycle}_A = 10 \text{ sec} \times 2\text{GHz} = 20\text{G cycles}$
 - By plugging it into the first equation,
 $6 \text{ seconds} = (1.2 \times 20\text{G cycles}) / f$

$$f_B = 4\text{GHz}$$

#Instructions and CPI

- The performance equation does not include any reference to the number of instructions needed to run a program
- Since computer executes instructions to run programs, the execution time must depend on the number of instructions executed
- Execution time is the number of instructions executed multiplied by the average time per instruction

$$\text{CPU Time} = \text{CPU clock cycles} \times T$$

$$\text{CPU clock cycles} = \# \text{ instructions} \times \text{Avg. clock cycles per inst (CPI)}$$

$$\text{CPU Time} = \# \text{ insts} \times \text{CPI} \times T = \# \text{ insts} \times \text{CPI} / f$$

#Instructions and CPI

$$\text{CPU Time} = \# \text{ insts} \times \text{CPI} \times T = \# \text{ insts} \times \text{CPI} / f$$

- **#insts** is determined by
 - How efficient your program is
 - How good the ISA is
 - How efficient machine code the compiler generates
- **CPI** is determined by program languages, compilers, ISA, and your CPU design (microarchitecture)
 - For example: sequential vs pipeline implementations
- **f** is determined by your CPU design (microarchitecture) and semiconductor technology
 - Critical path between flip-flops determines the clock frequency
 - Advanced semiconductor technology (45nm, 32nm, 22nm etc) would increase the clock frequency

CPI Example

- There are 2 computers (Computer A and Computer B). Their CPUs implement the same ISA, and use the same compiler to compile application programs. But microarchitectures are different.
 - Computer A has a **clock cycle time of 250ps** and **CPI of 2.0** when running a program
 - Computer B has a **cycle time of 500ps** and **CPI of 1.2** when running the same program
- Which is faster, and by how much?

$$\text{CPU Time} = \# \text{ insts} \times \text{CPI} \times T = \# \text{ insts} \times \text{CPI} / f$$

What is the **execution time** to run the program in **Computer A**?

$$\# \text{ insts} \times \text{CPI} (2.0) \times 250 \text{ ps} = \# \text{ insts} \times 500 \text{ ps}$$

What is the **execution time** to run the program in **Computer B**?

$$\# \text{ insts} \times \text{CPI} (1.2) \times 500\text{ps} = \# \text{ insts} \times 600 \text{ ps}$$

So, **A is faster!**

How much? = $\text{performance}_A / \text{performance}_B = \text{exetime}_B / \text{exetime}_A = 600\text{ps} / 500\text{ps} = 1.2$

Computer A is 20% faster than computer B

CPI in More Detail

- If different instructions take different numbers of cycles (assume that we have n different instructions),

CPU Time = **Clock cycles** X clock cycle time (T)

$$\text{Clock Cycles} = \sum_{i=1}^n (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

CPI Example

- Suppose that there is one computer (Hardware designer supplied CPIs in orange), and there are 2 compilers to compile an application program.
 - The compiler A generated the machine code of sequence 1
 - The compiler B generated the machine code of sequence 2
- Which compiler is better for the application program?

Instructions	A	B	C
CPI	1	2	3
Instruction count in sequence 1	2	1	2
Instruction count in sequence 2	4	1	1

Sequence 1:

- Clock cycles
 $= 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
- Avg. CPI $= 10/5 = 2.0$

Sequence 2:

- Clock cycles
 $= 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
- Avg. CPI $= 9/6 = 1.5$

Performance Summary

CPU Time = # insts X CPI X clock cycle time (T) = # insts X CPI / f

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
 - **Algorithm** affects the instruction count
 - **Programming language** affects the instruction count and CPI
 - **Compiler** affects the instruction count and CPI
 - **Instruction set architecture** affects the instruction count, CPI, and T (f)
 - **Microarchitecture** (Hardware implementation) affect CPI and T (f)
 - **Semiconductor technology** affects T (f)

SPEC CPU Benchmark

- Benchmarks are programs used to measure performance
 - Supposedly typical of actual workload
- Standard Performance Evaluation Corp ([SPEC](#)) is an effort funded and supported by a number of computer vendors to create standard sets of benchmarks for modern computer systems
 - SPEC89: In 1989, SPEC originally created a benchmark set focusing on processor performance
 - SPEC CPU2006 is the latest:
 - CINT2006 (integer) is for measuring and comparing compute-intensive integer performance
 - CFP2006 (floating-point) is for measuring and comparing compute-intensive floating-point performance

- **Backup Slides**

Some Basics

- Kilobyte (KB) – 2^{10} or 1,024 bytes
- Megabyte (MB) – 2^{20} or 1,048,576 bytes
- Gigabyte (GB) – 2^{30} or 1,073,741,824 bytes
- Terabyte (TB) – 2^{40} or 1,099,511,627,776 bytes
- Petabyte (PB) – 2^{50} or 1024 terabytes
- Exabyte (EB) – 2^{60} or 1024 petabytes