CS 271
Computer Architecture & Assembly Language

Lecture 20

Parallelism

Research & Innovation

Closing Remarks

3/10/22, Thursday

Oregon State University

Lecture Topics:

- Parallelism
- Research & Innovation
- Closing remarks

Parallelism

Hardware Parallelism (overview)

- Instruction-level parallelism
  - Pipeline
  - Cache
- Processor-level parallelism
  - Multiprocessor (multiple CPUs, common memory)
  - Multicomputer (multiple CPUs, each with own memory)

Pipelining

U-1	U-2	U-3	U-4	U-5
Instruction Fetch	Instruction Decode	Operand Fetch	Instruction Execute	Operand Store

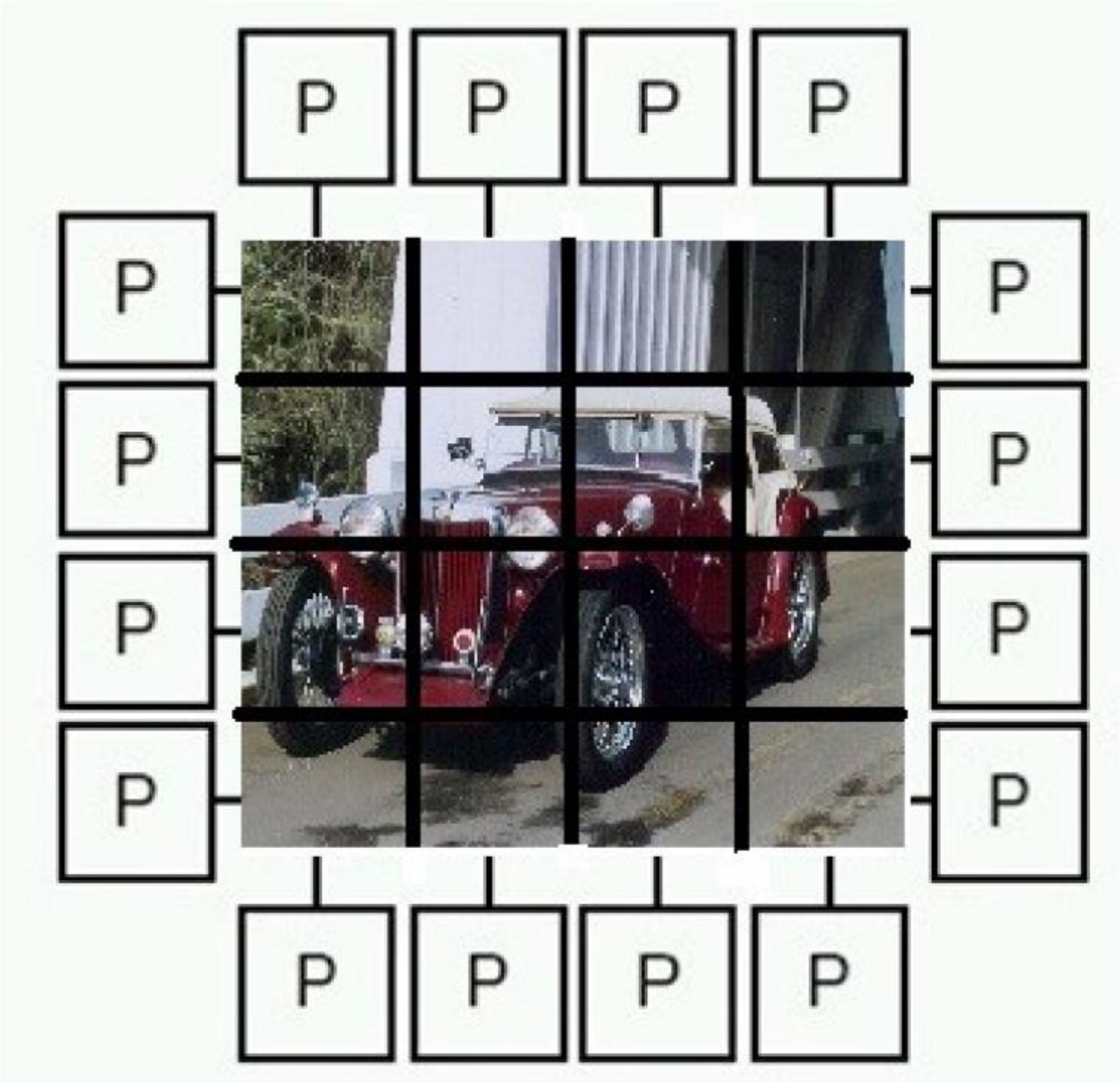
Instruction Sequence										
U-1	1	2	3	4	5	6	7	8	9	10
U-2		1	2	3	4	5	6	7	8	9
U-3			1	2	3	4	5	6	7	8
U-4				1	2	3	4	5	6	7
U-5					1	2	3	4	5	6
	T-1	T-2	T-3	T-4	T-5	T-6	T-7	T-8	T-9	T-10
Time →										

A 5-stage pipeline, showing which unit is processing each instruction number in each clock cycle.

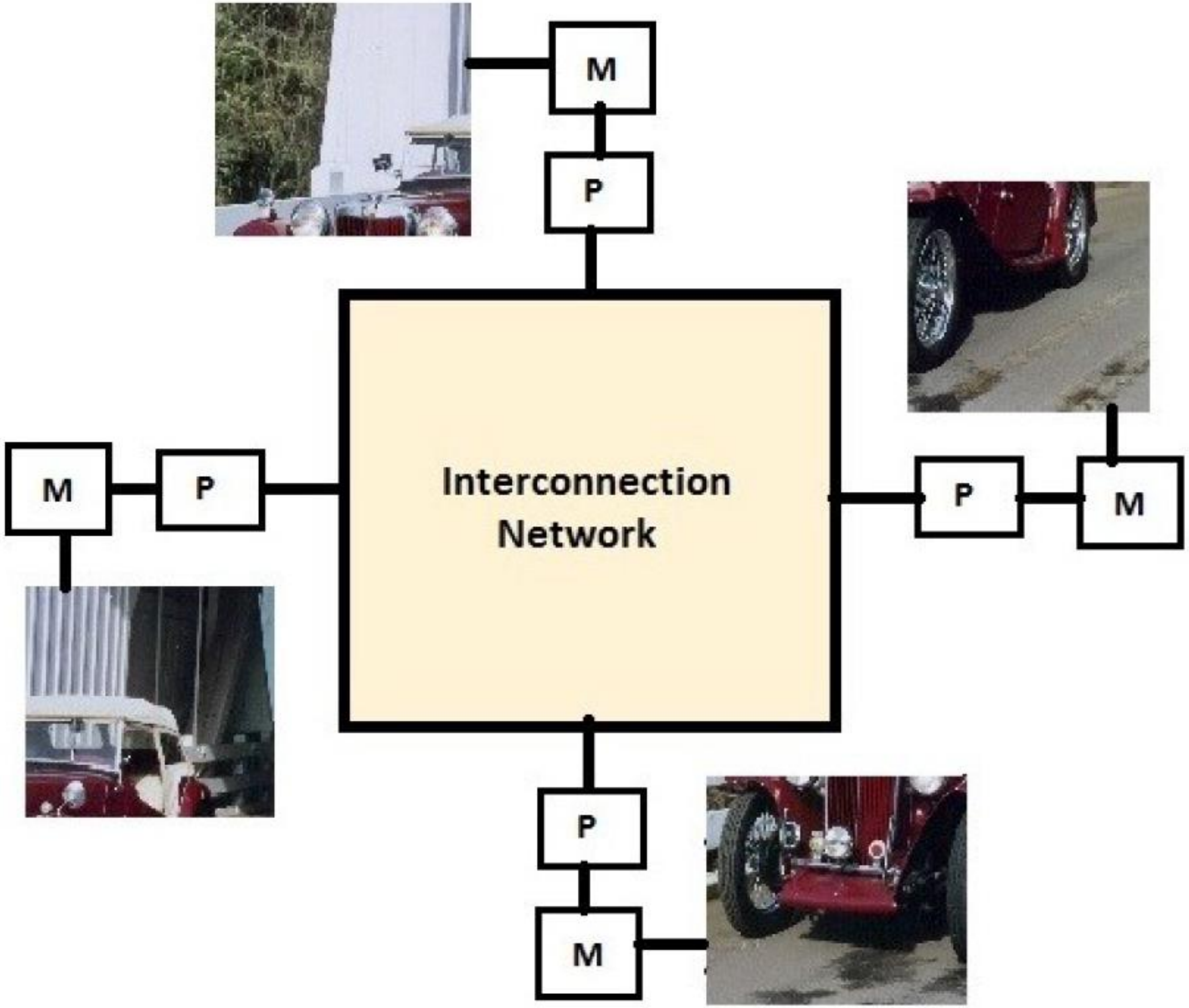
Instruction Caching

- Hardware provides area for multiple instructions in the CPU
  - Reduces number of memory accesses
  - Instructions are available for immediate execution
  - Might cause problems with decision, repetition, and procedure structures in programs

Multiprocessor Parallelism (shared memory)



Multicomputer Parallelism (distributed memory)



Comparisons

- Multiprocessor
  - Difficult to build
  - Relatively easy to program
- Multicomputer
  - Easy to build (given networking technology)
  - Extremely difficult to program
- Hybrid systems
  - Cloud computing

Interconnection Network

- Communication among processors
- Multiprocessor system
  - Communication through circuits/memory
- Multicomputer system
  - Communication through networking technologies
  - Packets (data, source/destination information, etc.)
  - Links, switches, interfaces, etc.

Software Parallelism

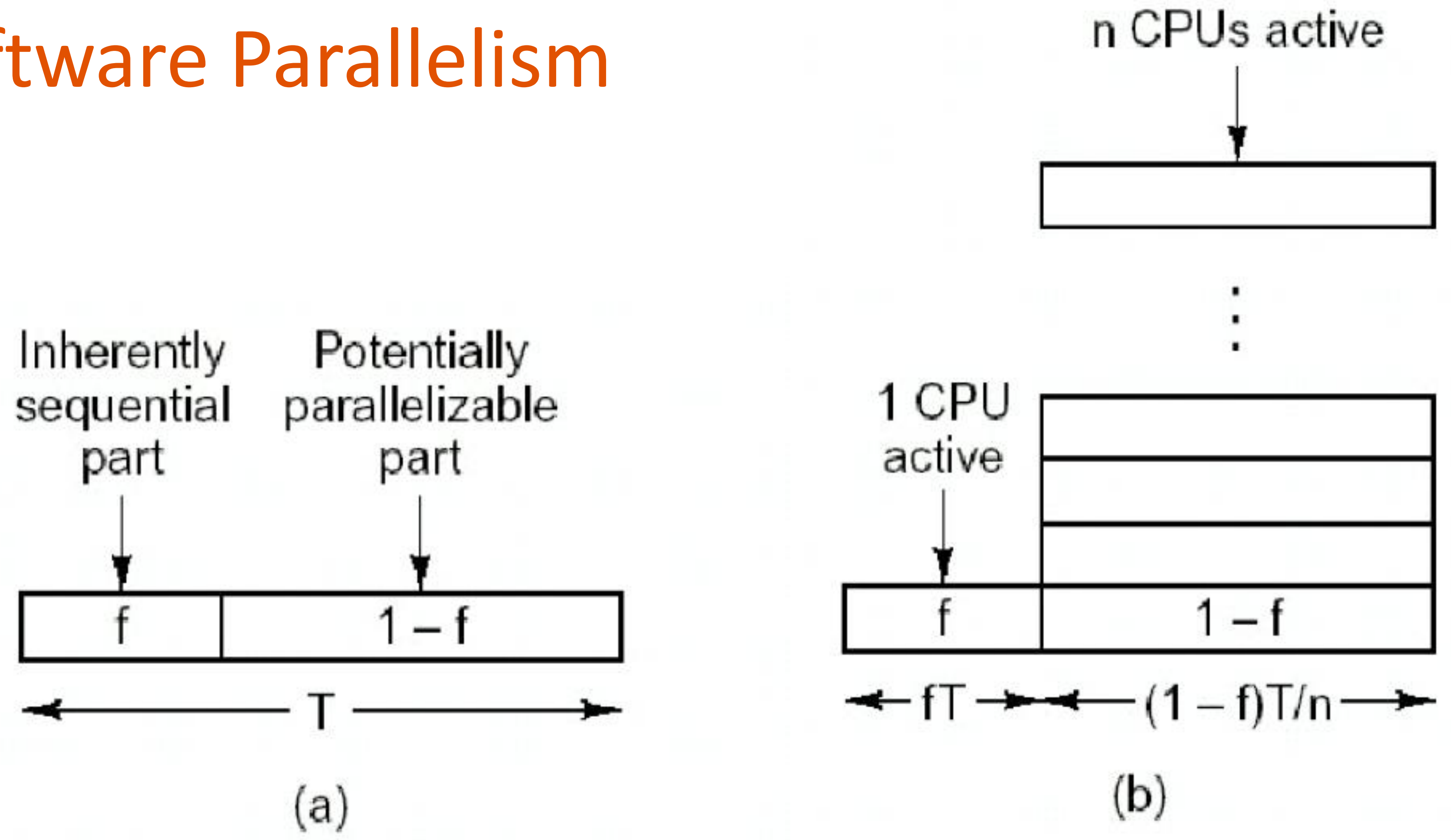
- Parallelizability of algorithms
  - Number of processors
  - Trade-offs and efficiency
  - Sequential/parallel parts
- Amdahl's Law
  - n = number processors
  - f = fraction of code that is sequential
  - t = time to process entire algorithm sequentially (one processor)

$$speedup = \frac{n}{1 + (n-1)f}$$

- Note: total execution time is

$$fT + \frac{(1-f)T}{n}$$

Software Parallelism



(a) A program has a sequential part and a parallelizable part

(b) Effect of running the parallelizable part on a multi-processor architecture

Software Parallelism

- Example:
- An algorithm takes 10 seconds to executes on a single 2.4G processor. 40% of the algorithm is sequential. Assuming zero latency and perfect parallelism in the remaining code, how long should the algorithm take on a 16 X 2.4G processor parallel machine?

$$speedup = \frac{n}{1+(n-1)f} = \frac{16}{1+.4x15} = \frac{16}{7}$$

- Therefore the expected time is
- 10 / (16 / 7) = 4.375 seconds
- Another way: (0.4 * 10) + (0.6 * 10) / 16

seq. + parallel

$$\frac{T}{speedup}$$

Software Parallelism

$$speedup = \frac{n}{1 + (n - 1) f}$$

- Assuming perfect scalability, what are the implication on Amdahl's law when n → ∞?

- speedup → 1/f (assuming f ≠ 0)
- Therefore, if f = 0.4, parallelism can never make the program run more than 2.5 times as fast

More Parallelism

- As a Computer Scientist, you will encounter parallel systems, parallel algorithms, parallel programming … everywhere.

- It is important to understand the fundamentals of computer hardware in order to make the best uses of parallelism

Research & Innovation

Parallel Computing Performance Depends on Hardware/Software

- Hardware
  - CPU speed of individual processors
  - I/O speed of individual processors
  - Interconnection network
  - Scalability

- Software
  - Parallelizability of algorithms
  - Application programming languages
  - Operating systems
  - Parallel system libraries

Hardware Parallelism

- CPU and I/O speed:
  - Same factors as for single-processor machines … plus:
- Interconnection network
  - Latency (wait time):
    - Distance
    - Collision / collision resolution
  - Bandwidth (bps)
    - Bus limitations
    - CPU and I/O limitations
- Scalability
  - Adding more processors affects latency and bandwidth

Software Parallelism

- Parallel system libraries
  - Precompiled functions designed for multiprocessing (e.g., matrix transformations)
  - Functions for control of communication (e.g., background printing)

- Application programming languages
  - Built-in functions for creating child processes, threads, parallel looping, etc.
  - Mostly imperative (e.g., C)

- Operating systems

Application of Parallelism

- Multi-user systems
  - Networks
  - Internet

- Speed up single processes
  - Chess example
  - Expert systems
  - Other AI applications

Research in Parallelism

- Parallelism is an extremely hot research area
  - Especially in parallel software systems, parallel algorithms, etc.
  - Knowledge of parallel architectures is useful.

Innovations

- ExtremeTech
  - Learn about the “bleeding edge”

- What’s going on with embedding computing?
  - Integration / Specialization

Be Confident…

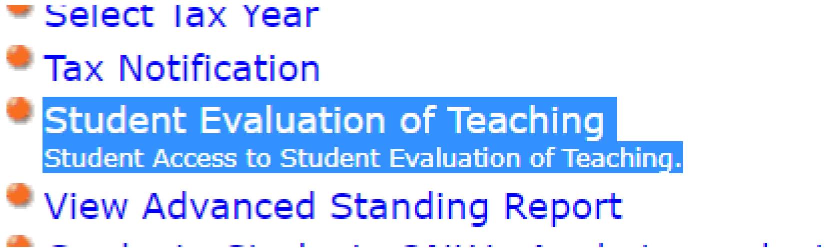
Now you are able to…

- Access and interpret binary data stored in memory.
- Illustrate the Instruction Execution Cycle.
- Create and analyze well-modularized assembly language programs utilizing decision, repetition, and procedure structures.
- Utilize a debugger to identify and correct bugs in assembly language programs.
- Illustrate the system stack as it is used for procedure calls and parameter passing.
- Understand the primary components of a modern computer architecture, and explain their function.

Final Remarks…

- Thank you so much for your commitment to this course

- Future improvements?
  - MyOSU → Student Records →



- ULA position
  - Contact me! And apply through: https://jobs.oregonstate.edu/postings/103887

Final Remarks…

- Submit all your work by the deadline
  - Weekly Summary 10
  - Quiz 4
  - Final Project

- Grade disputation:
  - By 3/20 6pm