EE/CSCI 451: Parallel and Distributed Computation

Lecture #2

8/20/2020

Viktor Prasanna

prasanna@usc.edu

ceng.usc.edu/~prasanna

University of Southern California

Outline

- High level view of Processor Core Organization

  Chapter 2.1, 2.2 in the text

  (Implicit parallelism)
  - Superscalar
  - Very Long Instruction Word (VLIW) processor
- Multi-core processor organization
- Implicit and Explicit Parallelism

Policies and Procedures

- All classes are recorded and accessible on Blackboard
- Recordings should be used appropriately (https://policy.usc.edu/scampus-part-c/)
- Specify your current time zone (and your background info.)
  - Please fill the Google form: https://forms.gle/yqxoUa7NC7bMq5yv6

Course Info. (1)

- When & Where
- Lecture: Tuesday, Thursday 3:30 – 4:50 PM, Online at: https://usc.zoom.us/j/92090885346?pwd=Ty9PZmFxL2dPVDR3ekNZNmdnVFJ0QT09
  - Meeting ID: 920 9088 5346
- Lab: Friday, 3:30 – 4:50 PM, Online at: https://usc.zoom.us/j/93866326251
  - Meeting ID: 938 6632 6251
- Office Hours
  - Tuesday 11:00-12:00 Noon
  - Thursday 10:00-11:00 AM
  - Meeting ID: usc.zoom.us/my/prasanna.zoom

Course Info. (2)

- TA & TA's office hours
  - Meng Yuan
    - Email: ymeng643@usc.edu
    - Office hours:
      - Friday 11:00 AM-1:00 PM or By appointment
      - Meet Yuan at: https://usc.zoom.us/j/8629150353
        - Meeting ID: 862 915 0353

Course Info. (3)

- Communicating with me
  - Visit during office hours via zoom
  - Via email prasanna@usc.edu
    - Subject field: EE451

Course Info. (4)

- Grading
  - Homework 10%
    - Homeworks must be done independently
    - **10% late penalty per day** will be assessed with no credit received after the third day
  - Programming Assignments 10%
  - Course Project 15%
  - Midterm I (Sep 25 in lab session, 2 hours) 20%
  - Midterm II (Oct 23 in lab session, 2 hours) 20%
  - Final Exam 25%

Course Info. (5)

- Lab
  - Time: 3:30 – 4:50 PM, Friday
  - Discussion of material covered in the lecture
  - Homework
  - Programming languages
  - Programming assistance
  - Course project discussion
  - Midterms (Sep 25 and Oct 23 in lab session, 330-530pm)
  - Course project presentation

Announcements

- Piazza
  - EE/CSCI 451 Parallel and Distributed Computation Fall 2020
  - Enroll via piazza.com/usc/fall2020/eecsci451
- Lecture notes (ppt) will be uploaded to Piazza one hour before the lecture
- Homework assignments will be uploaded to Piazza
- Please upload your completed homework on Blackboard

Midterms and Final

- Midterm I: 25th Sep in lab session (2 Hours)
- Midterm II: 23rd Oct in lab session (2 Hours)

- Final, as scheduled by the University
  - No makeup exams

Outline

- High level view of Processor Core Organization

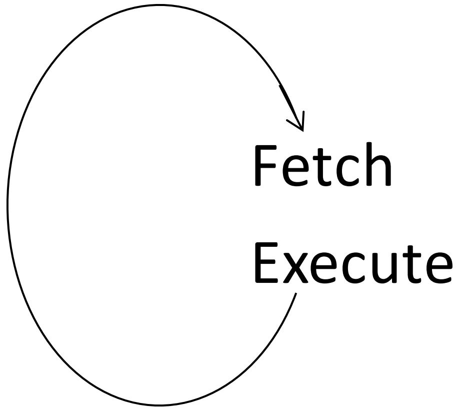
  Chapter 2.1, 2.2 in the text

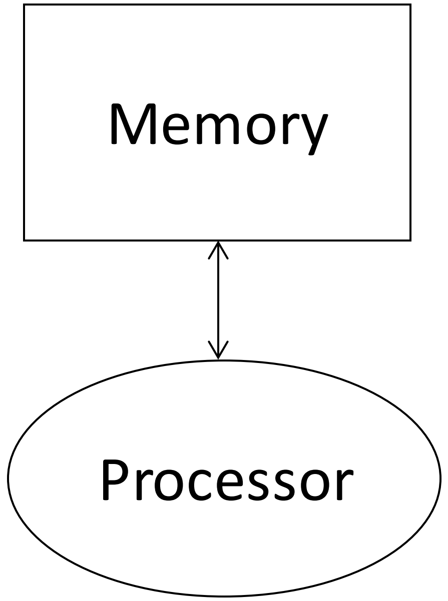
  (Implicit parallelism)

  - Superscalar
  - Very Long Instruction Word (VLIW) processor
- Multi-core processor organization
- Implicit and Explicit Parallelism

Uniprocessor (Serial Processor) RAM (Random Access Machine)

A simple view





Memory access = 1 cycle for any location
Execution = 1 cycle

Pipelined implementation of processor (1)

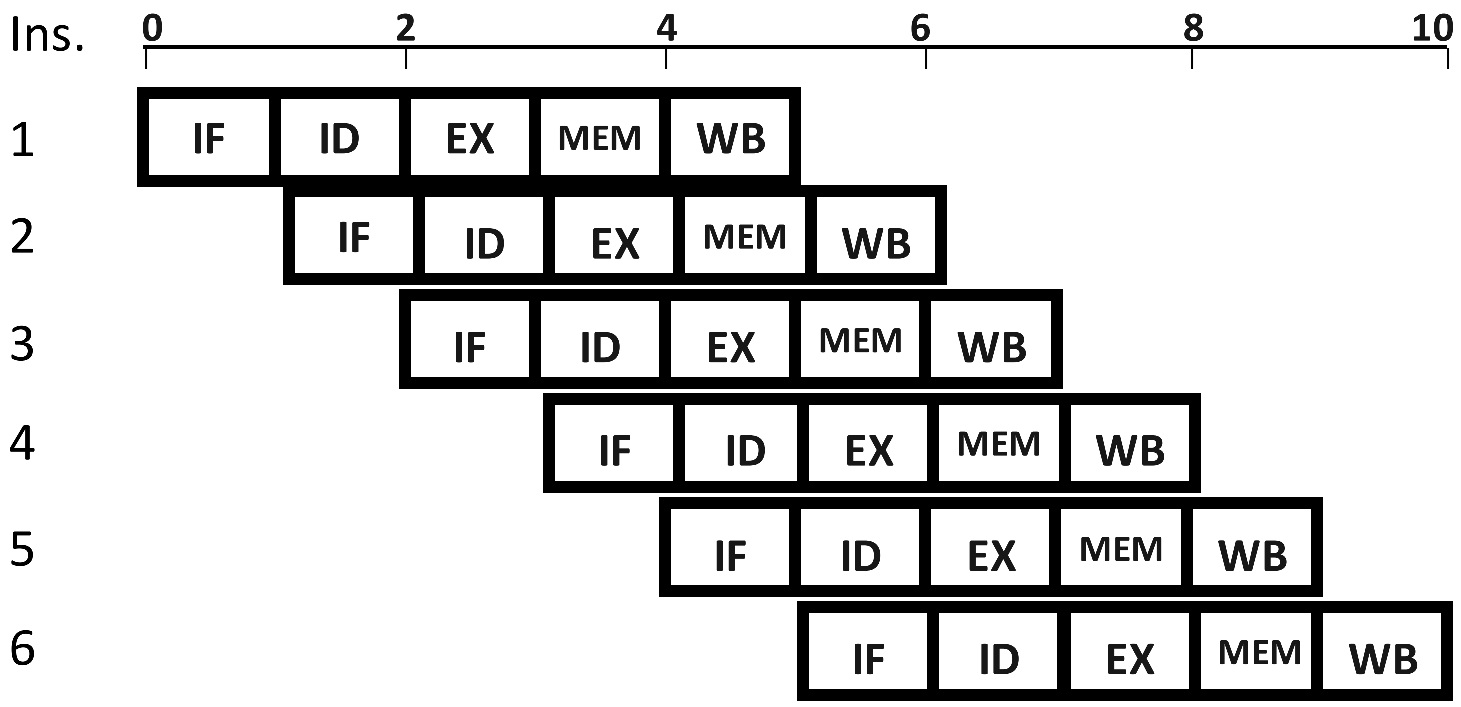
All (modern) processors are pipelined

Five stage pipeline (For example, Hennessey Patterson, EE 457)

$$\begin{tikzcd}
\boxed{\textbf{IF}} \arrow[r] & \boxed{\textbf{ID}} \arrow[r] & \boxed{\textbf{EX}} \arrow[r] & \boxed{\textbf{MEM}} \arrow[r] & \boxed{\textbf{WB}}
\end{tikzcd}$$

Instruction Fetch	**Instruction Decode/ Register file read**	**Execute/ Address calculation**	**Memory access**	**Write Back**

Pipelined implementation of processor (2)



1. Load R1, @1000
2. Load R2, @1008
3. Add R1, 100
4. Add R2, 104
5. Add R1, R2
6. Store R1, @2000

Total # of cycles to complete execution = 10

For executing n instructions, **5n ≥ total # of cycles ≥ n**

Ideally, total # of cycles = 5 + (n – 1)

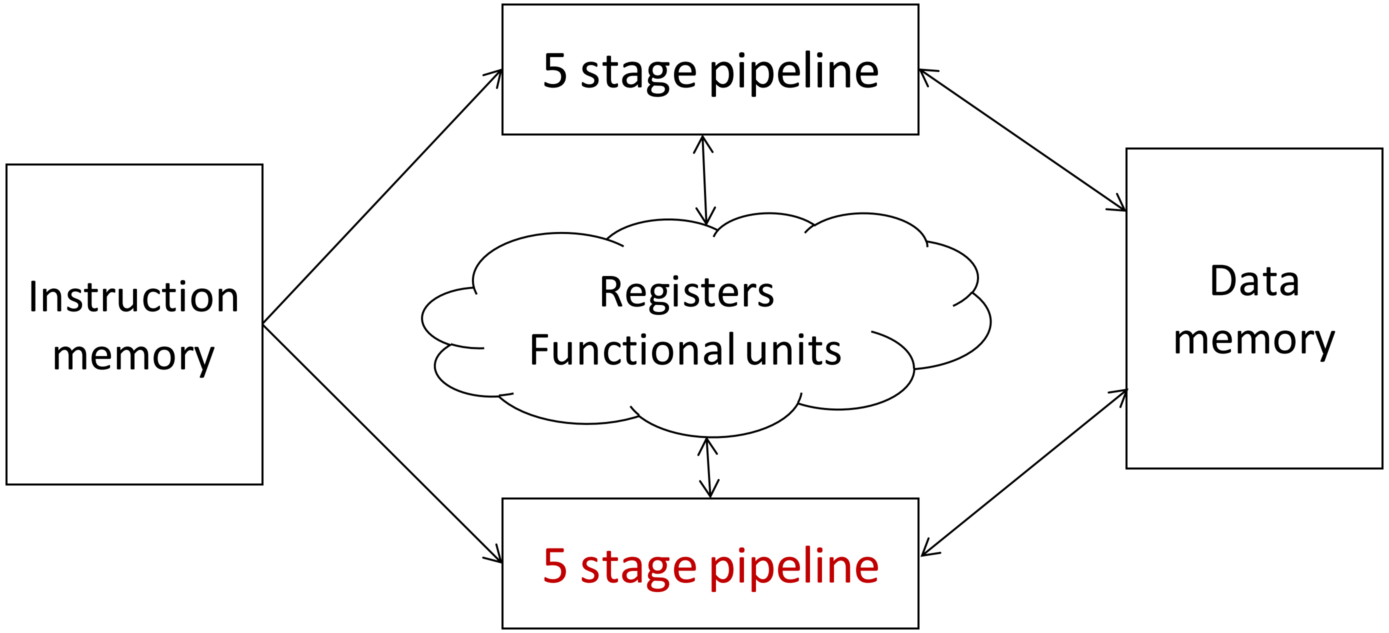
Simple Performance Model

- Instruction exec. rate = clock rate = peak performance ?
- Time to execute n ins. $\geq n$ / clock rate
- Example: 2 GHz processor core (single core)
- **Peak** performance = 2 x10^9 Ins. per second (MIPS rating)

Superscalar execution

Example: 2 pipelines (Sharing some hardware resources)

2× improvement?



Implicit Parallelism

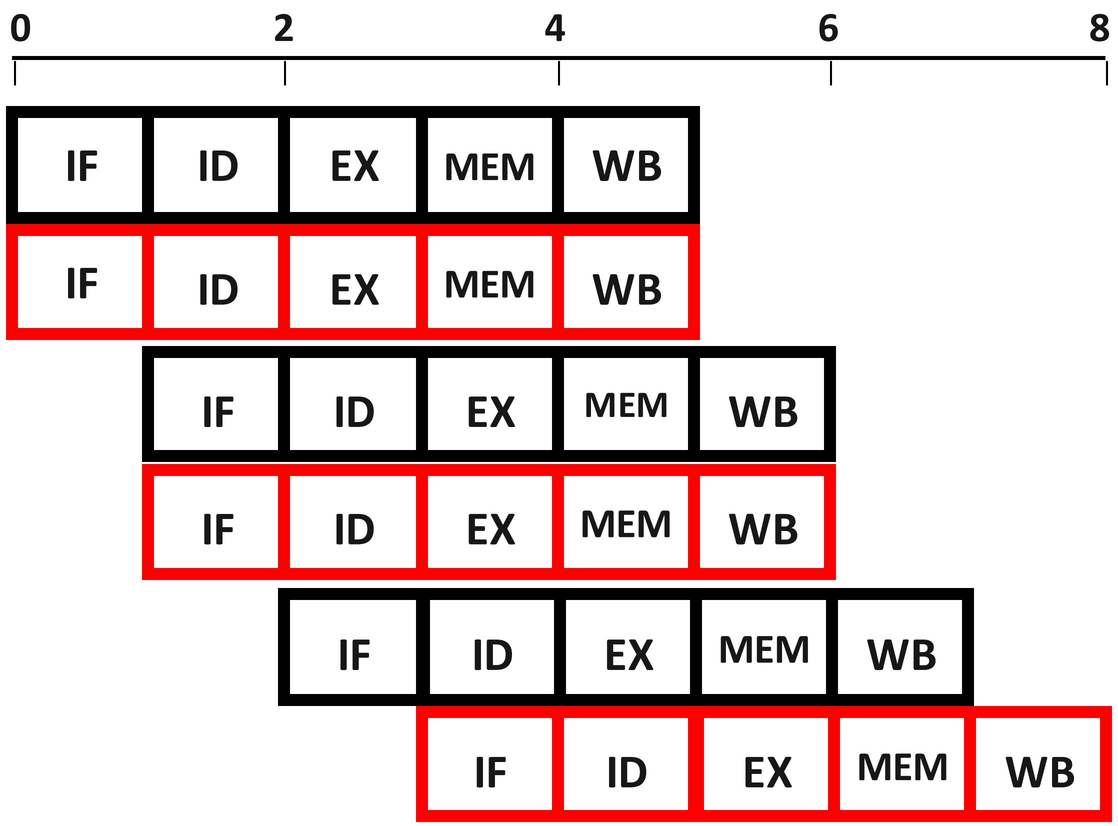
Implicit Parallelism

- 5 stage pipeline
  - 5n $\rightarrow$ 5 + n – 1 cycles

- k-way superscalar
  - 5 + n/k – 1 ?

2-way super scalar execution (almost 2 Ins. Issued per cycle)

1. Load R1, @1000
2. Load R2, @1008
3. Add R1, 100
4. Add R2, 104
5. Add R1, R2
6. Store R1, @2000



Total # of cycles = 8

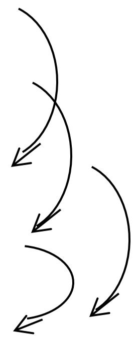
Note: Ins. 5 and 6 cannot be issued in the same cycle

Data dependencies (1)

Output of Ins i is an input to Ins k ($k > i$)
Potential problem: Ins k is close to Ins i **during** execution (during execution Ins k and Ins i are in the hardware pipeline concurrently)

Example: Assume Single Pipeline

1. Load R1, @1000
2. Load R2, @1008
3. Add R1, 100
4. Add R2, 104
5. Add R1, R2



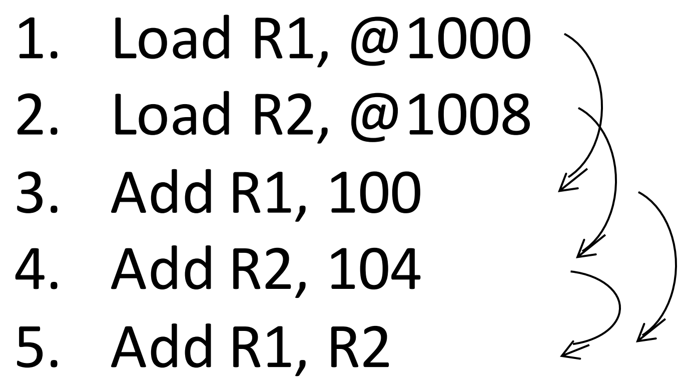
Simple solution = Stall the pipeline -> Performance loss
Hardware solution = Internal Forwarding

- Ex.: Dependency between Ins. 4 and 5 can be handled without stalling the pipeline.

Data dependencies (2)

- Data dependencies can have significant impact on performance
- We assumed a simple model of execution: **one cycle to access memory**
- In reality, main memory (DRAM) access latency is (very) high (50-100 cycles)

Example: (Execution of Ins 3 may have to be delayed by 10’s of cycles!)



1. Load R1, @1000
2. Load R2, @1008
3. Add R1, 100
4. Add R2, 104
5. Add R1, R2

Out of Order Issue

Example: 2-way superscalar

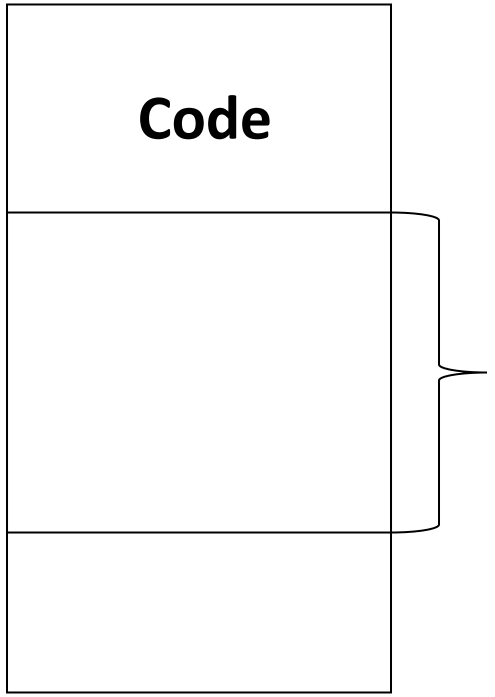
1. Load R1, @1000
2. Add R1, 100
3. Load R2, @1008
4. Add R2, 104

- Cannot issue Ins. 1, 2 simultaneously
- Cannot issue Ins. 3, 4 simultaneously

However, can do: Initiate Ins. 1, Ins. 3 in clock cycle 0

Initiate Ins. 2, Ins. 4 in clock cycle 1

Dynamic Instruction Issue



Window - Series of instructions

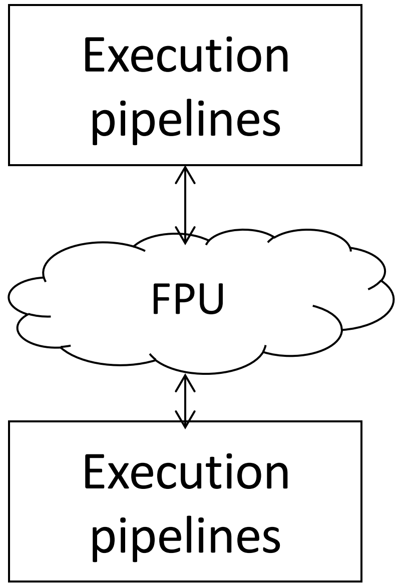
Identify those that can be executed in parallel

Note: Hardware does scheduling at run time

Way of exploiting instruction level parallelism

Resource dependency

Example: Suppose 2-way super scalar execution
one hardware Floating Point Unit (FPU)



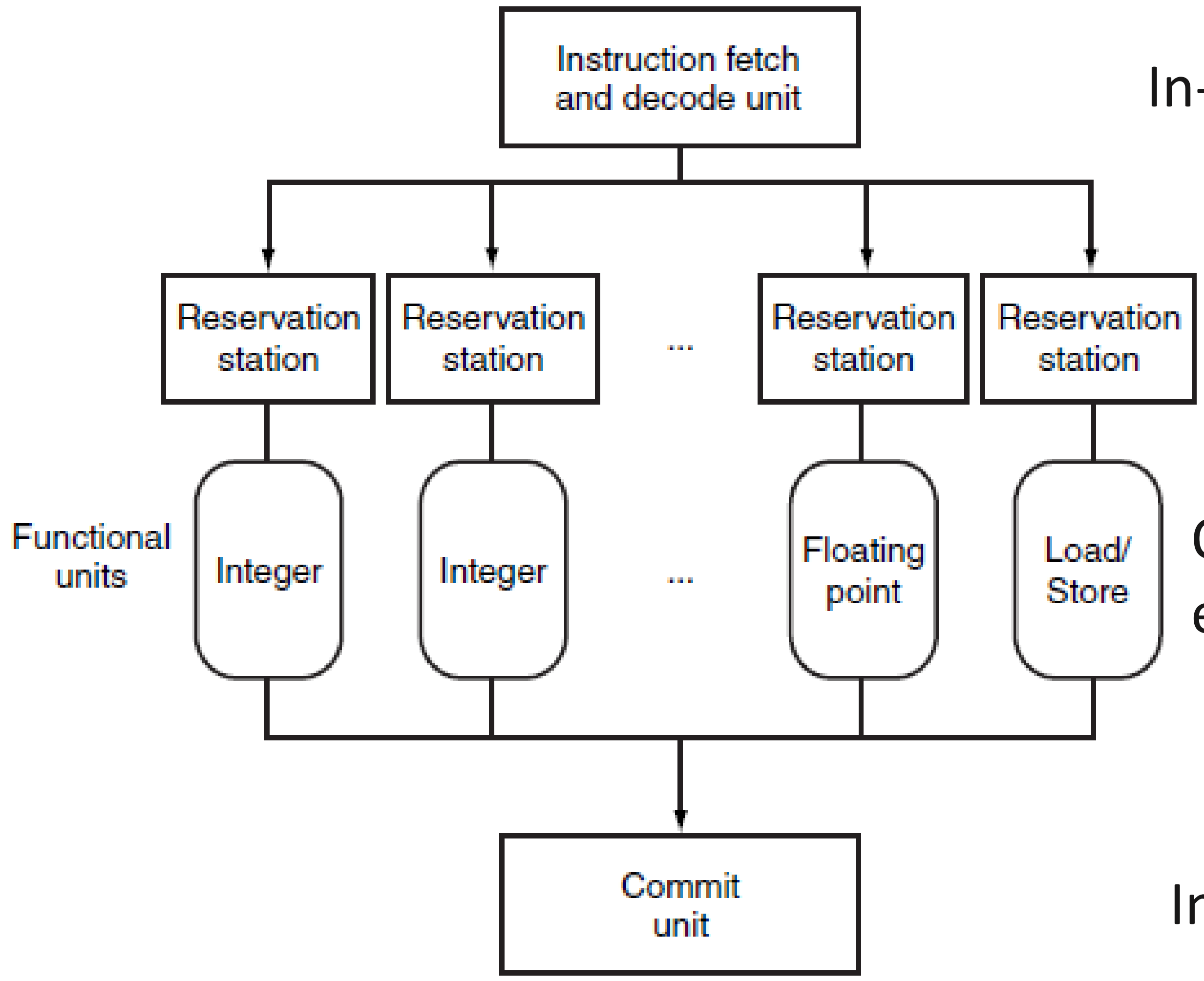
```
Load R1, @1000
Load R2, @1004
FP Mult R1, 100
FP Mult R2, 104
```

Execution hardware detects such dependencies and schedules the execution accordingly

Superscalar (1)

- Instruction-level parallelism
  - Multiple function units (Ex. arithmetic logic unit, bit shifter)
  - One or more instructions issued per clock
    - In-order issue
    - Out-of-order execution
    - In-order commit

Superscalar (2)



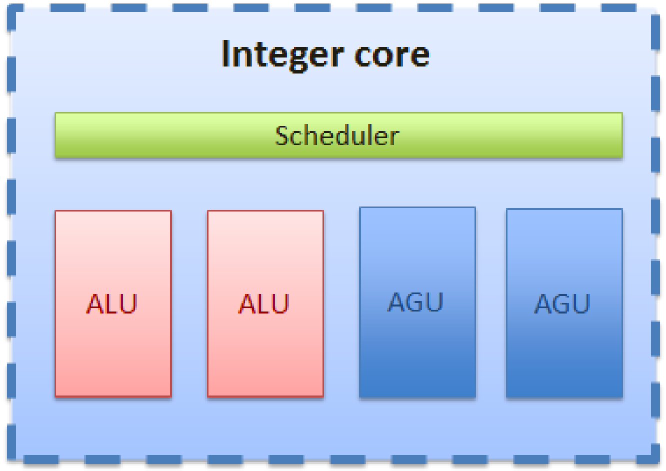
In-order issue

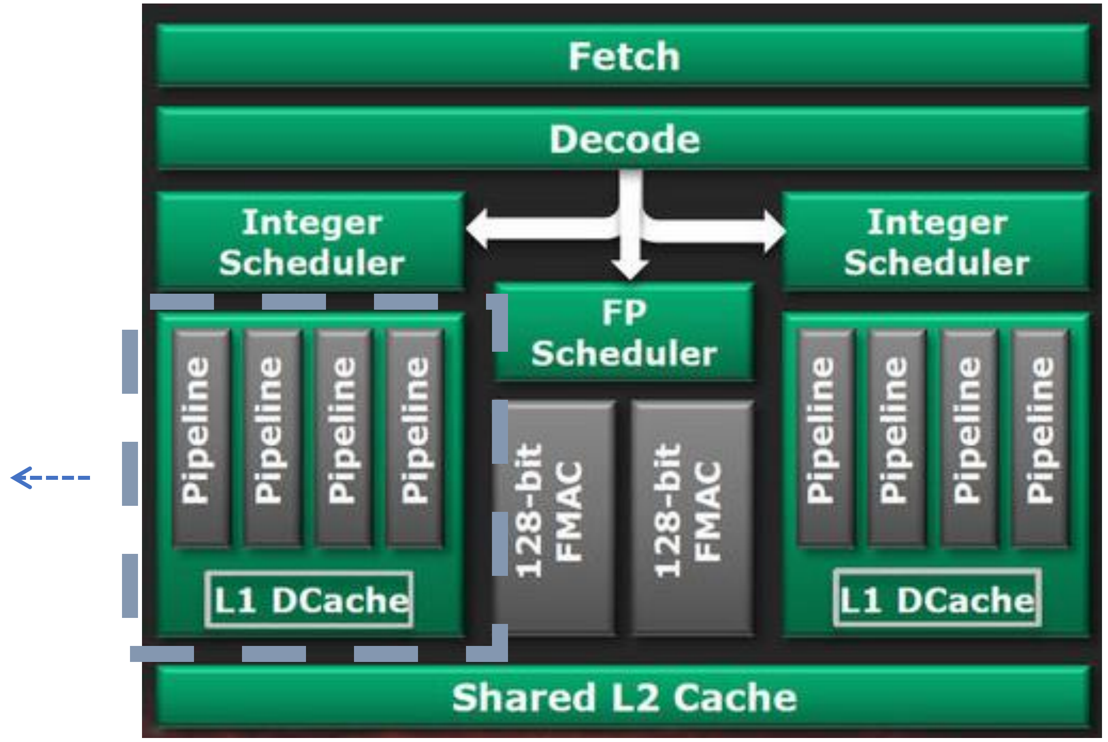
Out-of-order execute

In-order commit

Example Superscalar: Zen2 Architecture

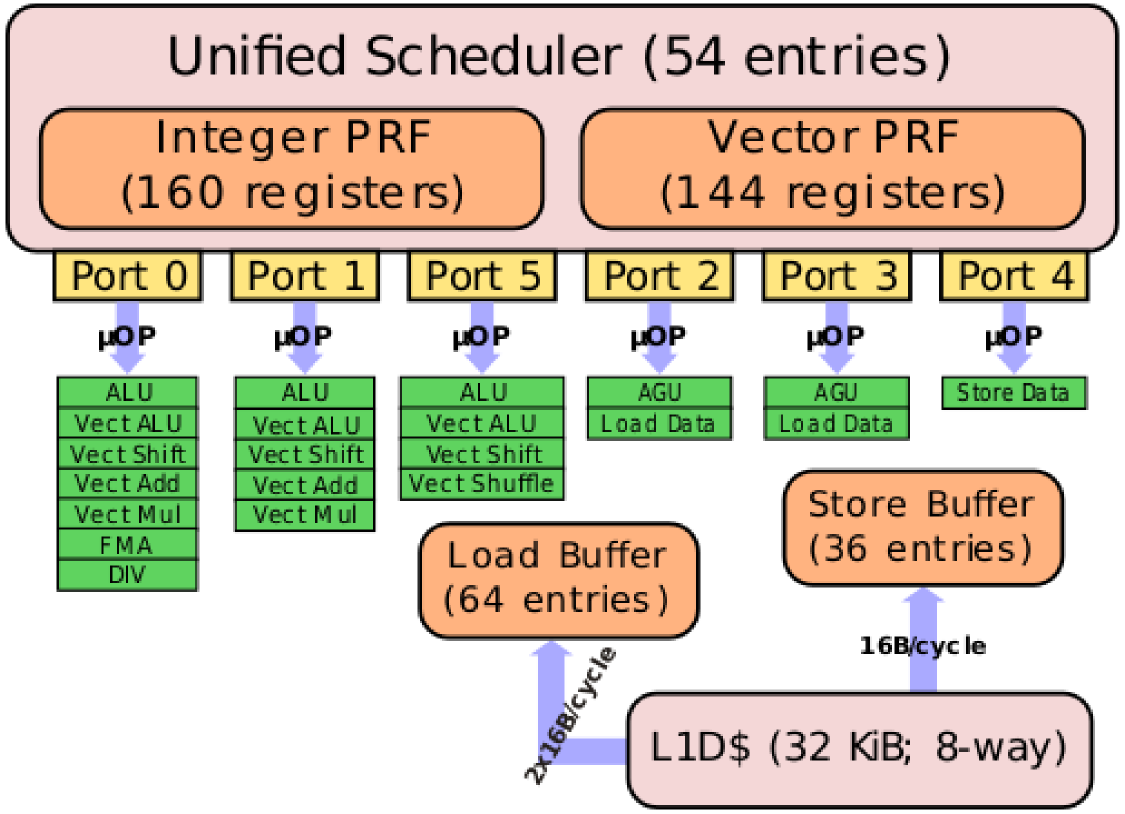
- Can handle 4 independent arithmetic and memory operations per clock for each integer core
- Issue bandwidth = 4





Example Superscalar: Sandy Bridge

- Microarchitecture by Intel
- Can handle up to 5 independent arithmetic and memory operations per clock for each core



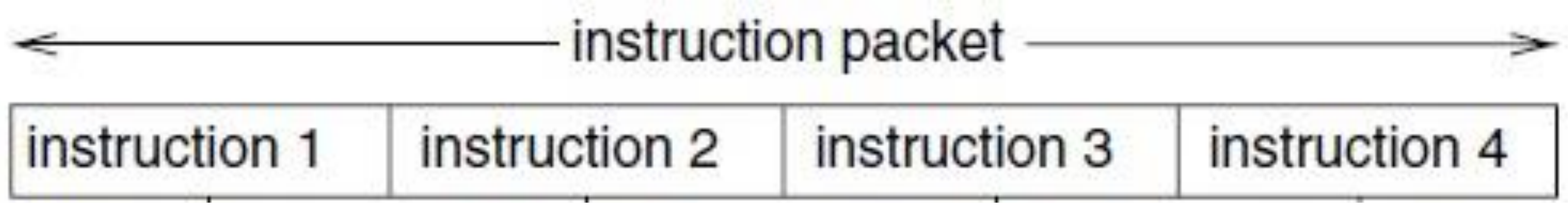
Very Long Instruction Word (VLIW) Processor

Superscalar processor

- Dynamic scheduling of Ins.
- Scheduling **hardware** is expensive
  - Dependence window size × # of pipelines

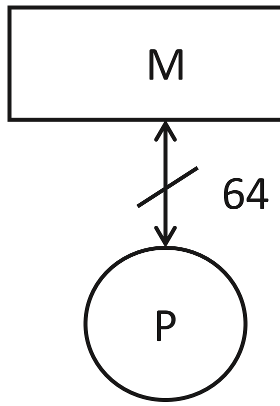
VLIW: Do scheduling at compile time (in **software**)

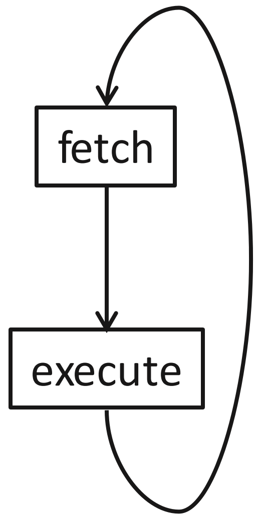
Specify parallelism as parallel activities (using long word instruction format)



Processor Execution Model

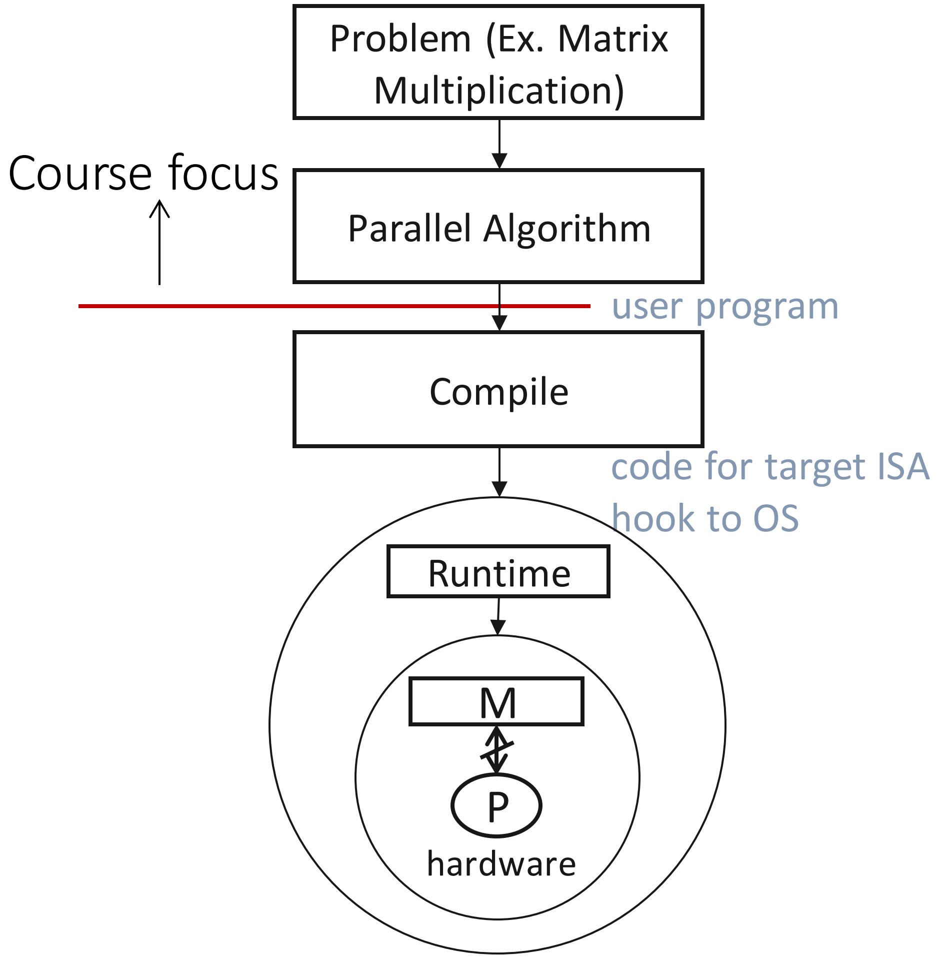
- SISD – Single Instruction Single Data
  - (Random Access Machine -- RAM)
- Instruction Set Architecture (ISA)
  - Abstract model of a computer, exposed to the users, complier





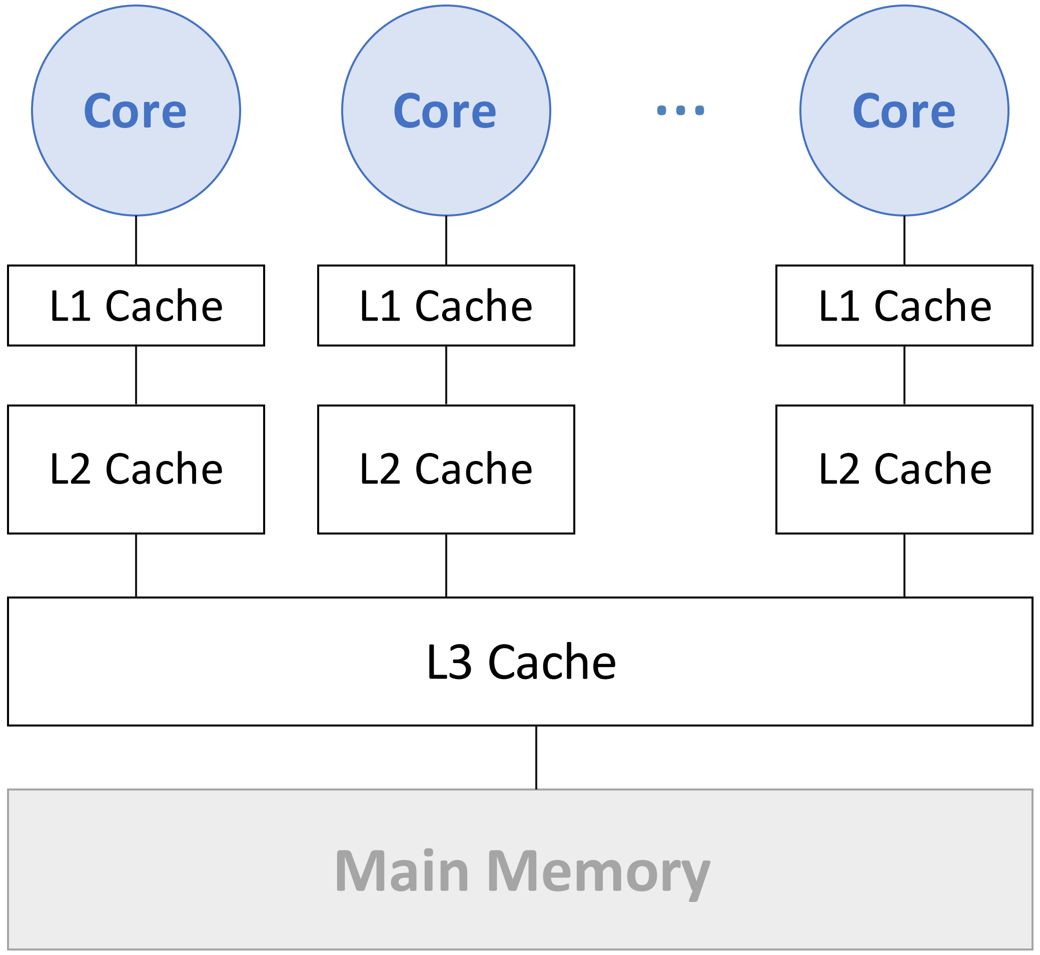
Designing and Executing a Parallel Program

- Performance metric
  - **Execution time**
- Execution time depends on
  - Compile time optimizations (Compiler level)
  - Execution time optimizations (Processor level—Implicit)
- Many optimizations are possible at architecture and hardware level
  (EE 457, EE 557)



Example Multi-core Processor (1)

- AMD Ryzen 9 3900X processor
  - 12 cores
  - 24 threads
  - 3.8 GHz
  - L3 cache: 64 MB
  - Memory: 2 Channels
  - PCIe: 24 Gen4 lanes



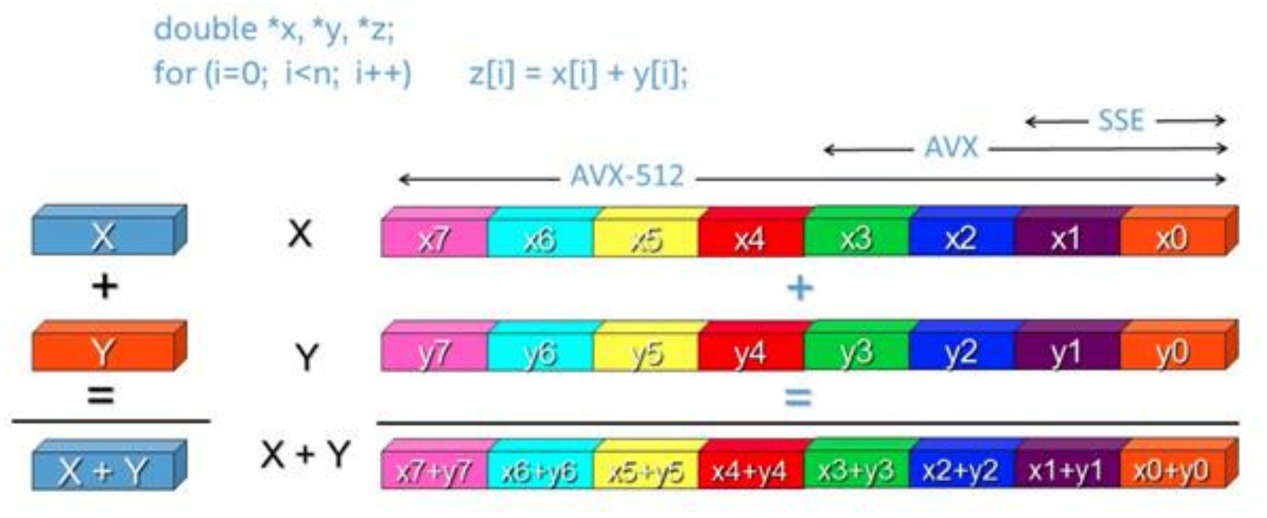
Multi-core Processor (2)

- Parallelism
  - Pipelined execution of Instructions (in each core)
  - Instruction-level parallelism (in each core)
  - Thread-level parallelism (across cores)

  - Data parallelism
  - Loop parallelism

Multi-core Processor (3)

- Data parallelism
  - Same operation on large data set
  - 512-bit Advanced Vector Extensions instructions (AVX-512) -- eight 64-bit or sixteen 32-bit integer operations per clock cycle



```
double *x, *y, *z;
for (i=0;  i<n;  i++)     z[i] = x[i] + y[i];
```

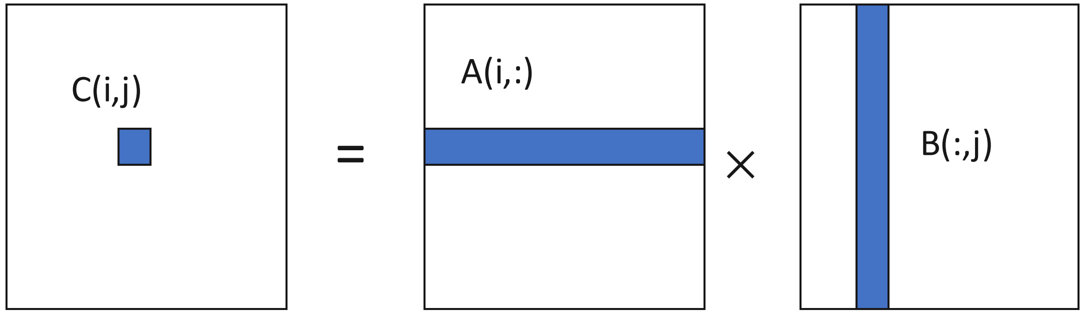
Multi-core Processor (4)

- Loop parallelism
  - Most execution time (of a serial program) is spent in loops
  - (90/10 rule)—90% of the time in 10% of the code
  - Extract parallel tasks from loops (for example, unrolling the loop)
    - Dynamically (hardware)
    - Statically by compiler (software)
  - Need to determine instruction dependence (data dependencies)
  - In C++: (compiler directives)
    - #pragma unroll
    - #pragma unroll(n)
    - #pragma nounroll

Explicit parallelism for Matrix Multiplication (1)

- Matrix Multiply
  $C = A \times B$ (A, B, C are $n \times n$ matrices with C[:,:] = 0)

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            $D(i,j,k) = A(i,k) \times B(k,j)$
            $C(i,j) = C(i,j) + D(i,j,k)$
```



$2n^3$ operations

Explicit parallelism for Matrix Multiplication (2)

- **Data parallelism**

  Compute A(i,k) × B(k,j) for various k in parallel within one instruction
- AVX-512 with 64-bit integer (for k=1,9,17,…):
  - Can do 8 ops/ins using 64 bit arithmetic
  - No data dependencies

A		A(i,k)	A(i,k+1)	A(i,k+2)	A(i,k+3)	A(i,k+4)	A(i,k+5)	A(i,k+6)	A(i,k+7)
B	×	B(k,j)	B(k+1,j)	B(k+2,j)	B(k+3,j)	B(k+4,j)	B(k+5,j)	B(k+6,j)	B(k+7,j)
D	=	D(i,j,k)	D(i,j,k+1)	D(i,j,k+2)	D(i,j,k+3)	D(i,j,k+4)	D(i,j,k+5)	D(i,j,k+6)	D(i,j,k+7)

- Compute C(i,j) from D(i,j,k)
- Explicit (parallel) program specification to exploit Implicit Parallelism (AVX-512) of the architecture

Explicit parallelism for Matrix Multiplication (3)

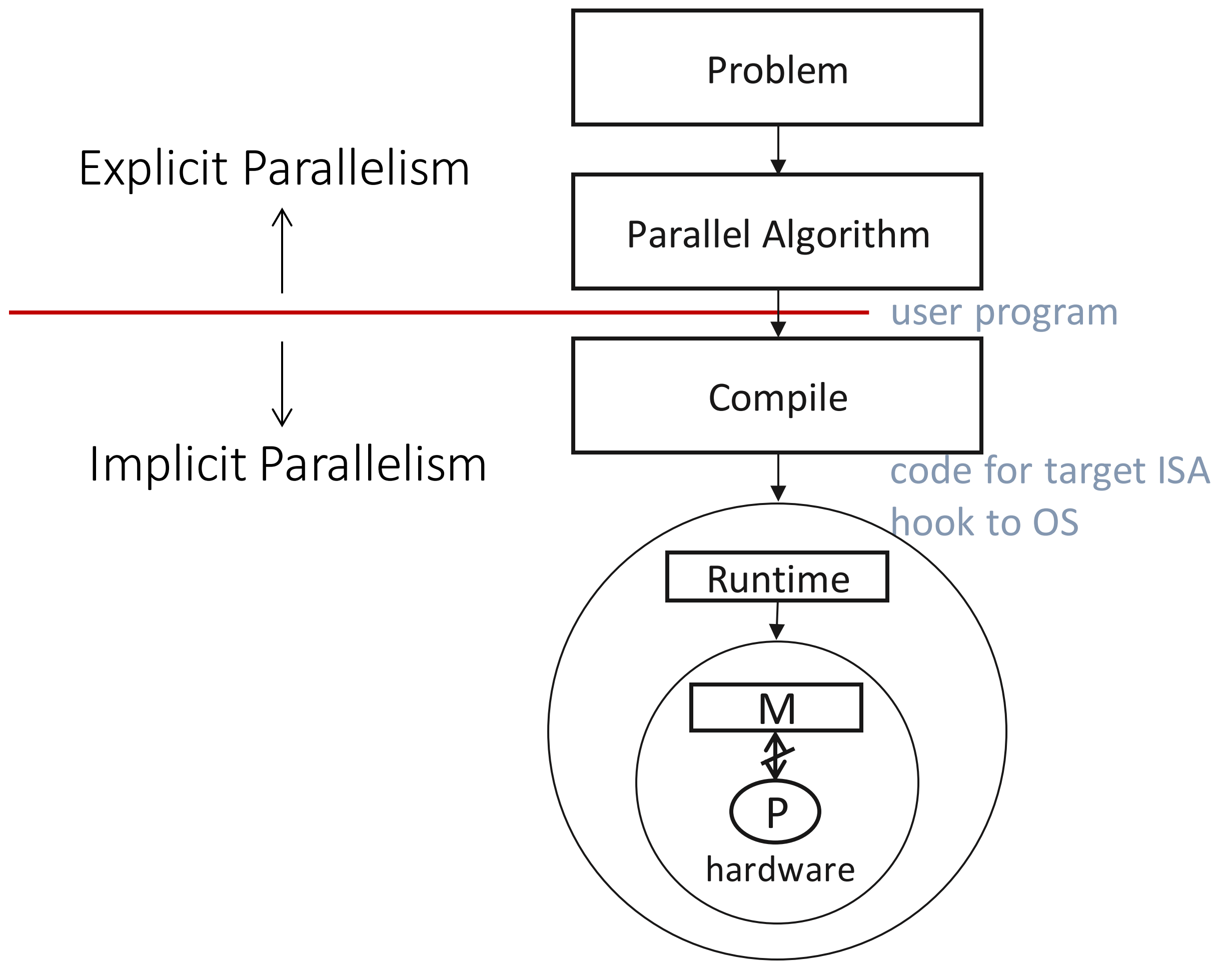
- **Loop parallelism**
  Compute A(i,k) × B(k,j) for various k in parallel using multiple threads
- Assume a two-thread multicore:
  Restructured loop program (by a compiler)

```
for i = 1 to n
    for j = 1 to n
        for k = 1,3,…,n-1
            D(i,j,k) = A(i,k) × B(k,j) //thread 0
            D(i,j,k+1) = A(i,k+1) × B(k+1,j) //thread 1
            C(i,j) = C(i,j) + D(i,j,k)
            C(i,j) = C(i,j) + D(i,j,k+1)
```

compute in parallel, no dependencies

thread coordination, data dependencies--can be handled

Summary (1)


Problem
Parallel Algorithm
Explicit Parallelism
user program
Implicit Parallelism
Compile
code for target ISA
hook to OS
Runtime
M
P
hardware

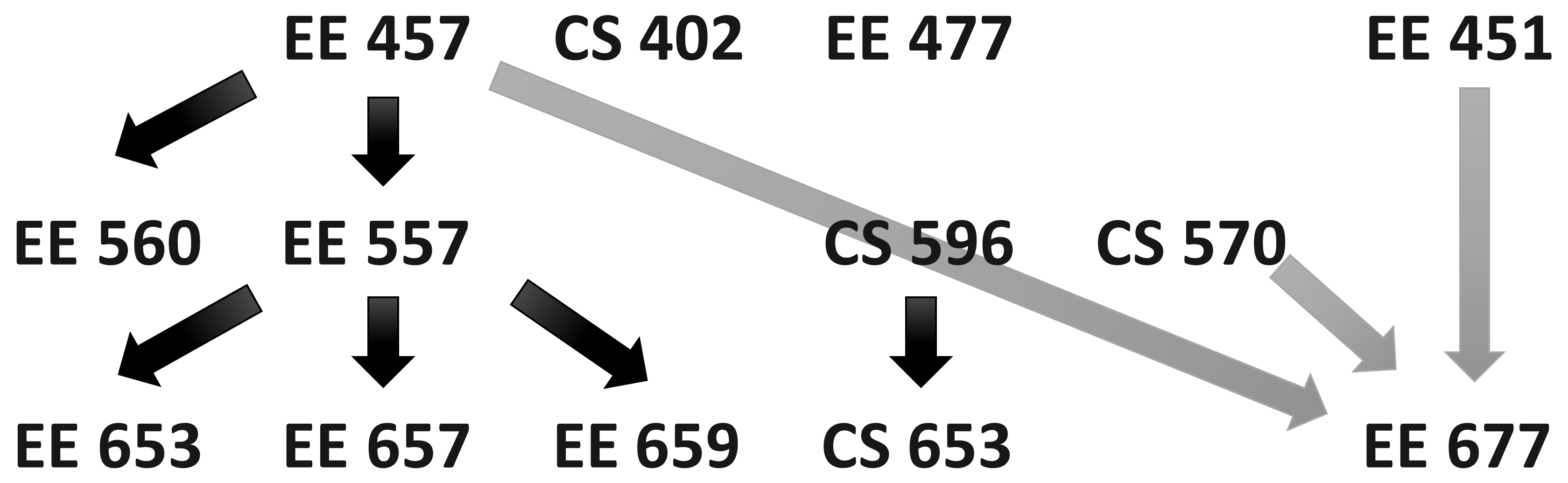
Summary (2)

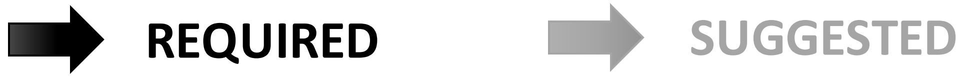
- SISD execution model
- Processor core organization
- Pipelined execution
- Data dependencies
- Multi-core architecture
- Explicit and Implicit Parallelism

- Course Focus: Explicit Parallelism

Backup Slides

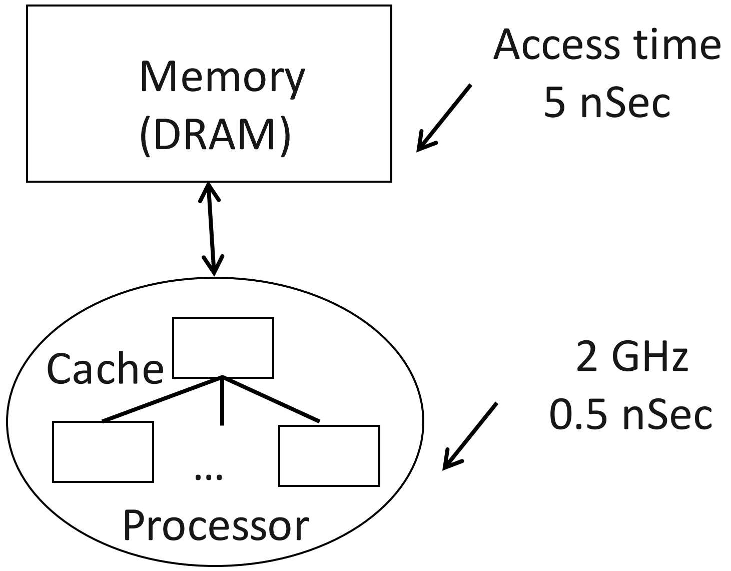
Course Dependencies





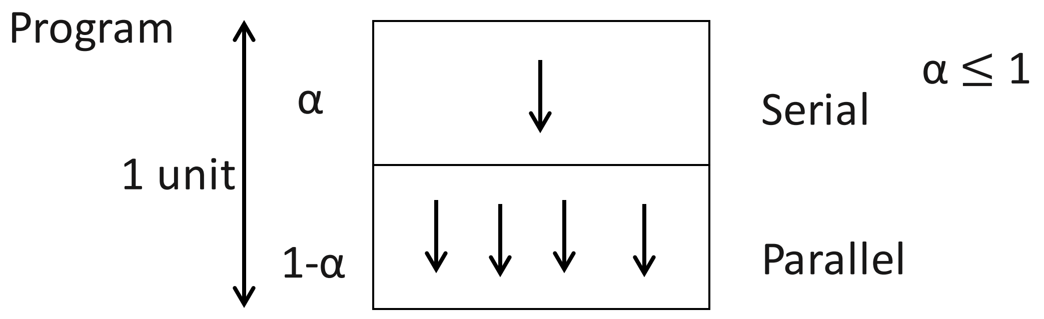
Challenges (1)

- Processor - Memory performance gap



Challenges (2)

- Limit to achievable speedup
  - Amdahl's Law



$$\text{Serial time} = 1,$$

$$\text{Best Parallel time} = \propto,$$

$$\text{Speedup} = \frac{\text{Serial time}}{\text{Parallel time}} = \frac{1}{\propto}.$$

Example: 50 % of the code is parallizable,

Speedup ≤ 2 (no matter how many processors are used)

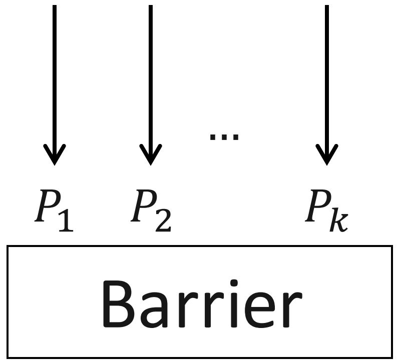
Challenges (3)

- Co-ordination

```
$Do\ in\ parallel$
        $P_1,$
        $P_2,$
        $\dots,$
        $P_k$
$end$
```

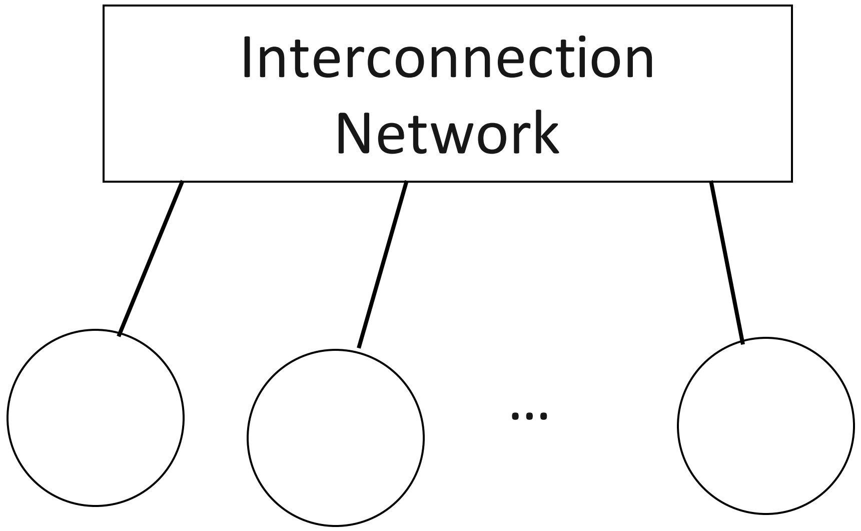
Synchronization

$overhead = ?$



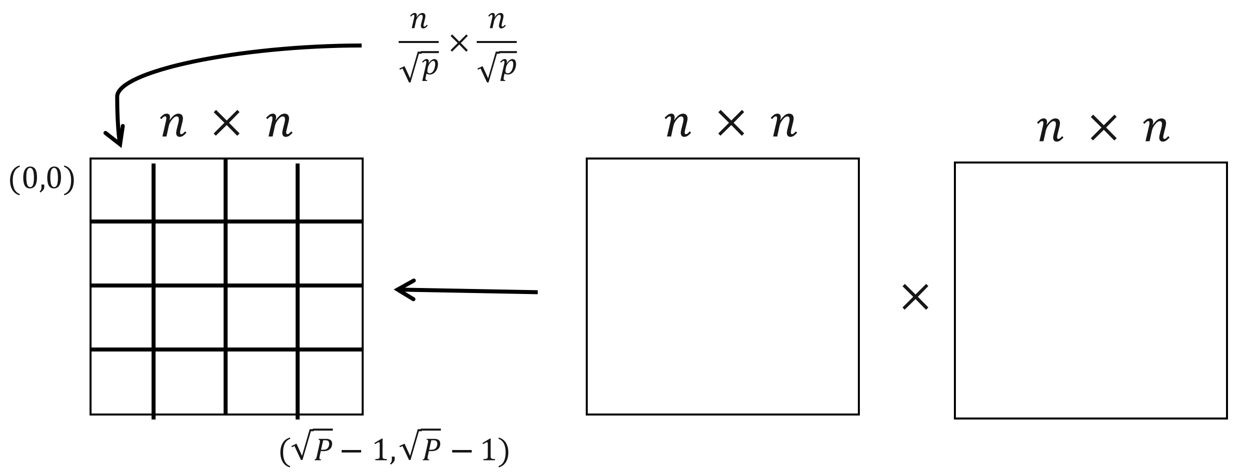
Challenges (4)

- Communication
  - Compute
  - Communicate
    - Wait?



Challenges (5)

- Load balance
  - Example: Matrix multiplication
    - $P_{ij} - compute\ output\ matrix\ ij,\ 0 \leq i, j < \sqrt{p}$
    - $p = number\ of\ processors$



$\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$
$n \times n$
(0,0)
$(\sqrt{P}-1, \sqrt{P}-1)$
$n \times n$
$\times$
$n \times n$

Challenges (6)

- Key performance metrics
  - $Speedup = \frac{Serial\ time}{Parallel\ time}$
  - Given $p > 1$ processors, can we get p fold improvement performance?
  - Scalability

Course Outline (1)

- Overview / Course Introduction
- Architectural Principles for Application Developers
  - Pipelined processor organization: data and control hazards, ILP, out of order execution, multithreading.
  - Memory systems: DRAM organization, cache organization.
  - Impact on software performance, locality, multithreading, prefetching.

Course Outline (2)

- Analytical Models for Parallel Systems
  - Architecture performance metrics: CPI, MIPS, SpecMark. Software performance benchmarks: Peak performance, sustained performance, LinPack, Bandwidth benchmarks.
  - Limits on achievable performance, Amdahl’s Law, scalability definitions, work optimality, Iso-efficiency function, Order notation.

Course Outline (3)

- Analytical Models for Parallel Systems(cont.)
  - Communication costs in parallel machines: start-up cost, throughput, latency. Routing mechanisms: packet routing, cut through, virtual channels. Modeling message passing and shared address space machines. Data layouts and graph embeddings.
  - Multi-core, many-core and GPU architectures, data parallel programming abstraction of GPUs.

Course Outline (4)

- PRAM and Data Parallel Algorithms
  - PRAM model of computation, Brent’s theorem, other models of computation, illustrative examples
  - Max, Scan operations
  - Recursive doubling, graph algorithms
  - Performance analysis, scalability, relation-ship to practical platforms
  - Sorting, FFT

Course Outline (5)

- Basic Communication Primitives
  - Broadcast and all-to-all, communication costs on various topologies
  - Personalized communication, Reduce, prefix sum, and scatter and gather
  - Graph embeddings

Course Outline (6)

- Message Passing Programming Model
  - Message passing abstraction, send receive primitives, blocking and non-blocking commands, collective operations
  - Illustrative examples: Canon’s algorithm, overlapping computation and communication, Odd--even merge sort, Review for Midterm 2
- Shared Address Space Programming Models
  - Pthreads , OpenMP, Illustrative examples

Course Outline (7)

- Parallel Dense Algebra
  - Matrix vector, matrix matrix computations
  - Parallel Dense Algebra
- Parallel Search and Sorting
  - Parallel search, illustrative example applications, throughput optimization, Multi-dimensional search, decision tree and decomposition
  - Sorting techniques, Bitonic sort, row-column sort, Mapping onto parallel architectures

Course Outline (8)

- Cloud, Big Data and Map Reduce
  - Cloud as a computing platform, Large data sets and organization
  - Discovery of knowledge, annotation of data
  - Computational and programming models for Big Data
  - Map Reduce as a parallel programming model, Hadoop, Spark
  - Illustrative examples from data science domain

Acknowledgement

- Lecture notes compiled over the years
- My research team
- Course TA and mentors during Spring ‘14, ‘15, ’16, and ‘17
- EE451 Students in Spring ‘14, ‘15, ‘16, and ‘17
- Introduction to Parallel Computing (2nd Ed.), Vipin Kumar, Ananth Grama
- ……

Research

- Research Projects
    - High Performance Networking
    - Big Data Analysis
    - Social Network Analysis
    - Energy Efficient Computing
- Webpages
    - ceng.usc.edu/~prasanna/
    - http://fpga.usc.edu/
    - http://pdc.usc.edu/
    - http://dslab.usc.edu/

Related Courses

- EE/CSCI 451 Introduction to Parallel and Distributed Computation (Prasanna)
- EE 454 Introduction to Systems-on-Chip (Bogdan)
- EE 457 Computer Systems Organization (Puvvada)
- CSCI 402 Operating Systems (Cheng)
- EE 557 Computer Systems Architecture (Annavaram/Dubois)
- EE 560L Digital System Design — Tools and Techniques (Puvvada)
- EE 532 Wireless Internet and Pervasive Computing (Hwang)
- EE 533 Network Processor Programming and Design (Cho)
- CSCI 558L Internetworking and Distributed System Laboratory (Cho)
- CSCI 570 Analysis of Algorithms (Shamsian/Adamchik)
- CSCI 596 Scientific Computing and Visualization (Nakano)
- EE 653 Advanced Topics in Microarchitecture (Dubois)
- CSCI 653 High Performance Computing and Simulations (Nakano)
- EE 659 Interconnection Networks (Pinkston)
- EE 657 Parallel and Distributed Computing (Dubios)
- EE 677 VLSI Architectures and Algorithms (Prasanna)

Sample course work with focus on Computer Architecture

- EE/CSCI 451 Introduction to Parallel and Distributed Computation (Prasanna)
- EE 454 Introduction to Systems-on-Chip (Bogdan)
- EE 457 Computer Systems Organization (Puvvada)
- CSCI 402 Operating Systems (Cheng)
- EE 557 Computer Systems Architecture (Annavaram/Dubois)
- EE 533 Network Processor Programming and Design (Cho)
- EE 599 Cyber-Physical Systems (Bogdan)
- CSCI 570 Analysis of Algorithms (Shamsian/Adamchik)
- EE 653 Advanced Topics in Microarchitecture (Dubois)
- EE 659 Interconnection Networks (Pinkston)
- EE 677 VLSI Architectures and Algorithms (Prasanna)

Sample course work with focus on Hardware, embedded systems

- EE/CSCI 451 Introduction to Parallel and Distributed Computation (Prasanna)
- EE 454 Introduction to Systems-on-Chip (Bogdan)
- EE 457 Computer Systems Organization (Puvvada)
- EE 477 MOS VLSI Circuit Design (Nazarian)
- EE 577a VLSI System Design (Pedram/Nazarian)
- EE 577b VLSI System Design (Pedram)
- EE 560L Digital System Design — Tools and Techniques (Puvvada)
- EE 532 Wireless Internet and Pervasive Computing (Hwang)
- EE 533 Network Processor Programming and Design (Cho)
- EE 599 Cyber-Physical Systems (Bogdan)
- CSCI 570 Analysis of Algorithms (Shamsian/Adamchik)
- EE 677 VLSI Architectures and Algorithms (Prasanna)

Sample course work with focus on parallel and scientific computing

- EE/CSCI 451 Introduction to Parallel and Distributed Computation (Prasanna)
- EE 454 Introduction to Systems-on-Chip (Bogdan)
- EE 457 Computer Systems Organization (Puvvada)
- CSCI 402 Operating Systems (Cheng)
- EE 557 Computer Systems Architecture (Annavaram/Dubois)
- EE 542 Internet and Cloud Computing (Hwang)
- CSCI 570 Analysis of Algorithms (Shamsian/Adamchik)
- CSCI 596 Scientific Computing and Visualization (Nakano)
- CSCI 653 High Performance Computing and Simulations (Nakano)

Sample course work with focus on distributed systems and big data

- EE/CSCI 451 Introduction to Parallel and Distributed Computation (Prasanna)
- EE 450 Introduction to Computer Networks (Zahid/Touch)
- EE 542 Internet and Cloud Computing (Hwang)
- CSCI 402 Operating Systems (Cheng)
- CSCI 555 Advanced Operating Systems (Govindan)
- CSCI 567 Machine Learning (Sha/Liu)
- CSCI 570 Analysis of Algorithms (Shamsian/Adamchik)
- CSCI 585 Database Systems (Raghavachary/Ghyam)
- CSCI 657 Advanced Distributed Systems (Lloyd)