

บทที่ 7

การประมวลผล/คอมพิวเตอร์แบบขนาน (Parallel Processing/Computer)

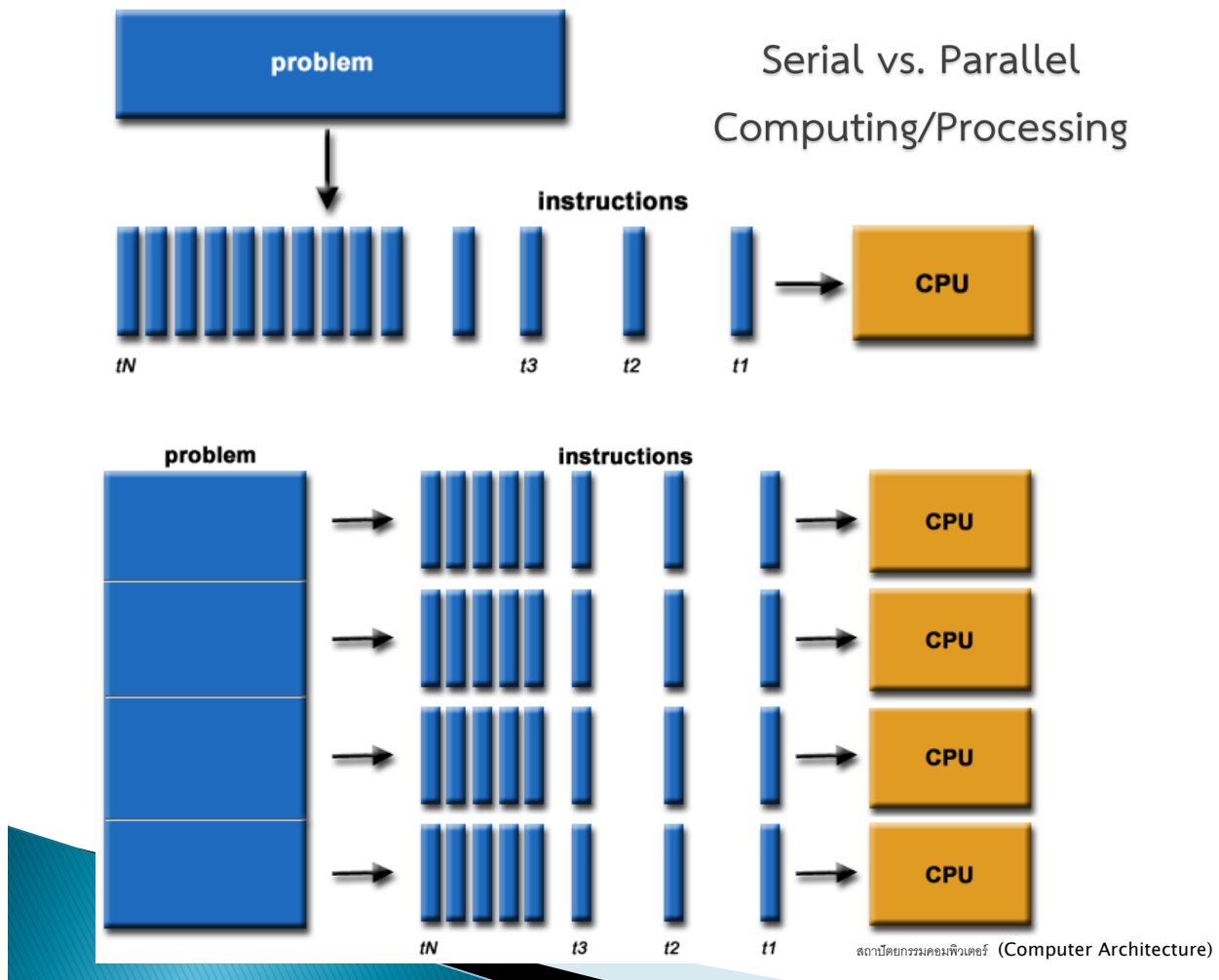
ผศ.ดร. สุรินทร์ กิตติธรกุล

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

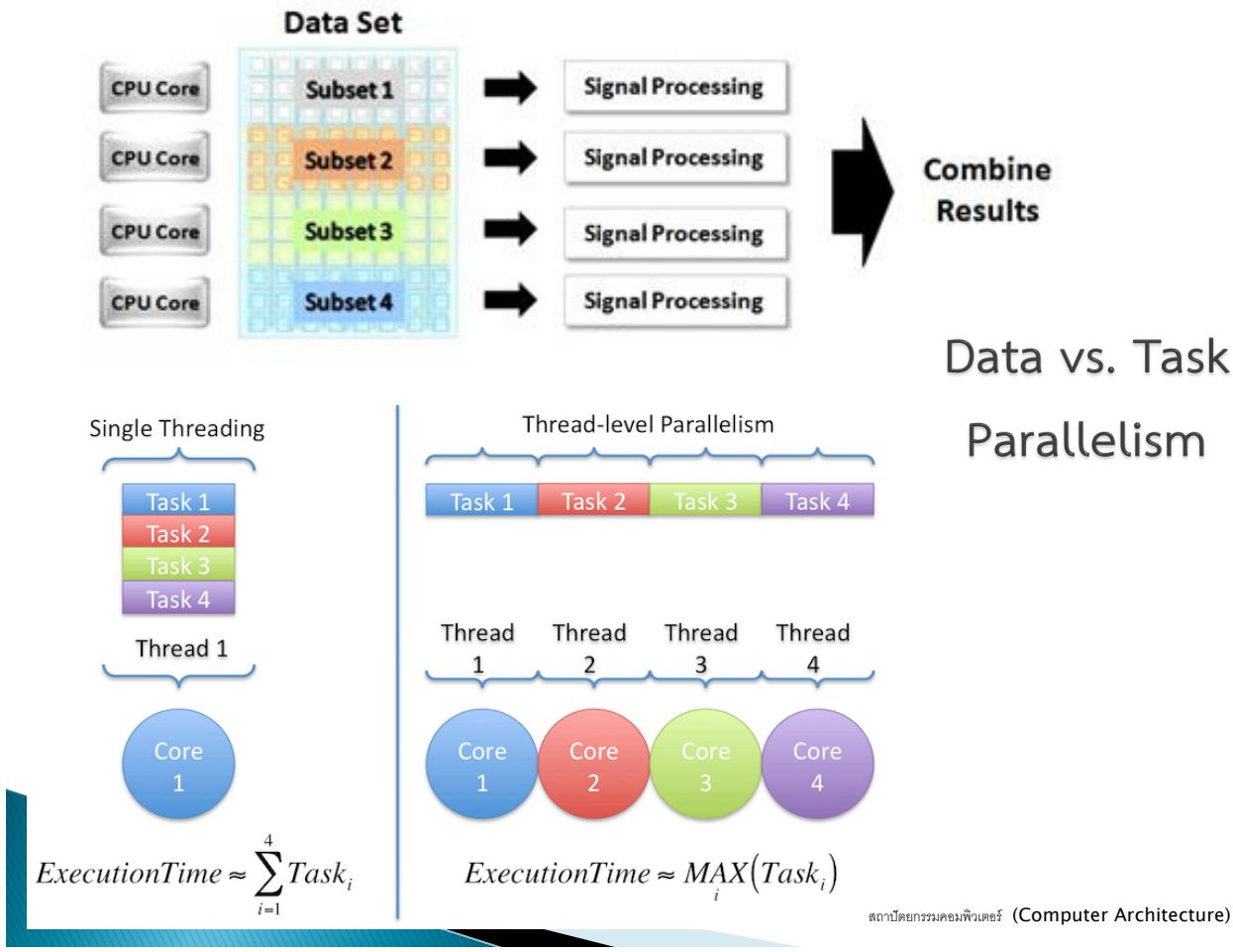
วัตถุประสงค์

- ▶ เพื่อให้เข้าใจความแตกต่างระหว่างการคำนวณแบบอนุกรมและแบบขนาน
- ▶ เพื่อให้เข้าใจการพัฒนาซอฟต์แวร์แบบขนาน
- ▶ เพื่อให้รู้จักคอมพิวเตอร์แบบขนานชนิดต่างๆ
- ▶ เพื่อให้สามารถแยกแยะเครื่องคอมพิวเตอร์แบบขนานชนิดต่างๆ ตามหลักการกายวิภาคของ Flynn (Flynn's Taxonomy)
- ▶ เพื่อให้เข้าใจกฎของ Amdahl (Amdahl's Law)



การพัฒนาซอฟต์แวร์แบบขนาน

- ▶ การพัฒนาซอฟต์แวร์ให้ทำงานแบบขนาน เป็นเรื่องไม่ง่าย
- ▶ ปัญหาแบ่งออกเป็น 2 ชนิด คือ
 - ปัญหาที่มีการวน返ของข้อมูล (Data Parallelism)
 - ปัญหาที่มีการวน返ของการทำงาน (Task Parallelism)
- ▶ ความยาก (Difficulties) คือ
 - การแบ่งภาระงาน (Partitioning)
 - การประสานงาน (Coordination)
 - การสิ้นเปลืองเวลาในการสื่อสาร (Communications overhead)



คอมพิวเตอร์แบบขนาน (Parallel Computer)

- ▶ เป้าหมาย: เชื่อมต่อเครื่องคอมพิวเตอร์หลาย台 เครื่องเพื่อสมรรถนะที่สูงขึ้น
 - โดยใช้หลักการมัลติโปรเซสเซอร์ (Multiprocessor)
 - เพื่อรับภาระงานที่เพิ่มขึ้น (Scalability), การใช้งานได้ตลอดเวลา (Availability), และประสิทธิภาพการใช้พลังงาน
- ▶ การทำงานแบบขนานระดับจ็อบหรือโปรเซส (Job-level or process-level parallelism)
 - เพิ่มทรัพุทให้กับจ็อบที่อิสระต่อกัน
- ▶ โปรแกรมการทำงานแบบขนาน (Parallel processing program)
 - โปรแกรมหนึ่งตัวสามารถทำงานพร้อมกันบนโปรเซสเซอร์หลายตัวหรือบนโปรเซสเซอร์ชนิดหลายคอร์ (Multicore microprocessors)

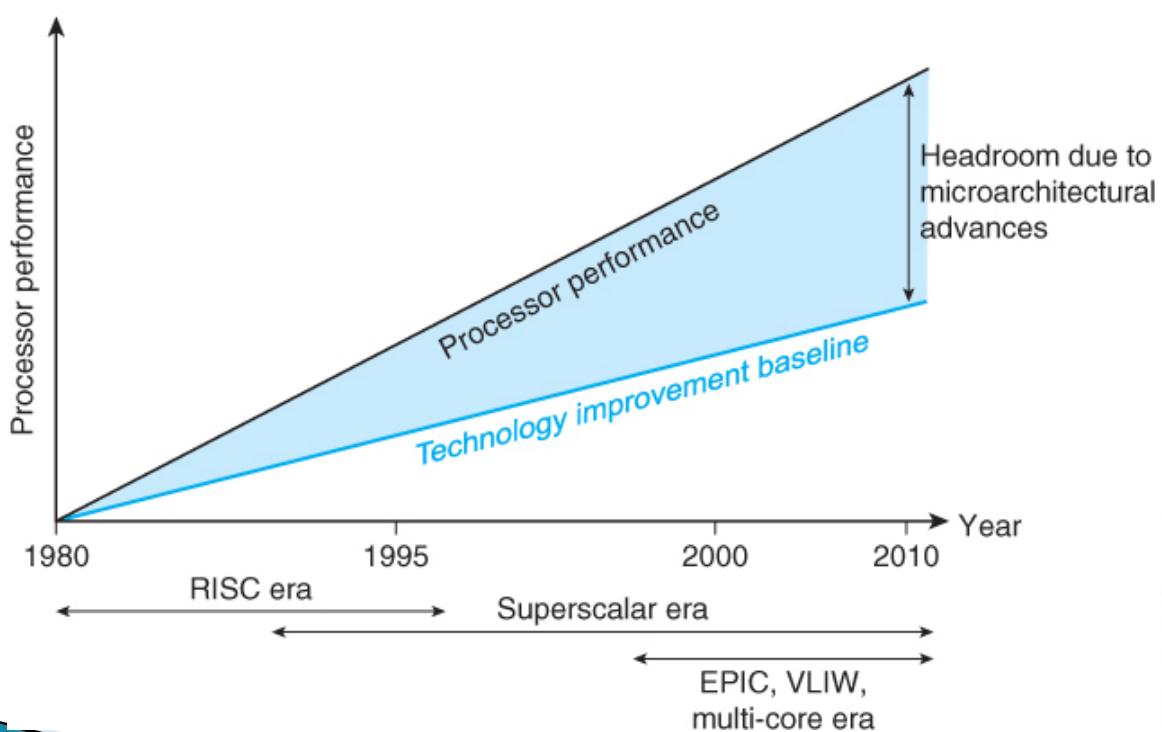
นิยาม

- ▶ ฮาร์ดแวร์ (Hardware)
 - แบบอนุกรม (Serial): เช่น โปรเซสเซอร์ Pentium 4 (Single Core)
 - แบบขนาน (Parallel): เช่น Xeon e5345 สี่คอร์ (quad-core)
- ▶ ซอฟต์แวร์ (Software)
 - แบบตามลำดับ (Sequential): เช่น การคูณเมทริกซ์ (matrix multiplication)
 - แบบคอนเคอเรนท์ (Concurrent): เช่น ระบบปฏิบัติการ (operating system)
- ▶ ซอฟต์แวร์ แบบตามลำดับ / แบบคอนเคอเรนท์ สามารถทำงานบน ฮาร์ดแวร์แบบอนุกรมและแบบขนานได้
 - ความท้าทาย คือ การใช้ประโยชน์จากฮาร์ดแวร์ให้มากที่สุด

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

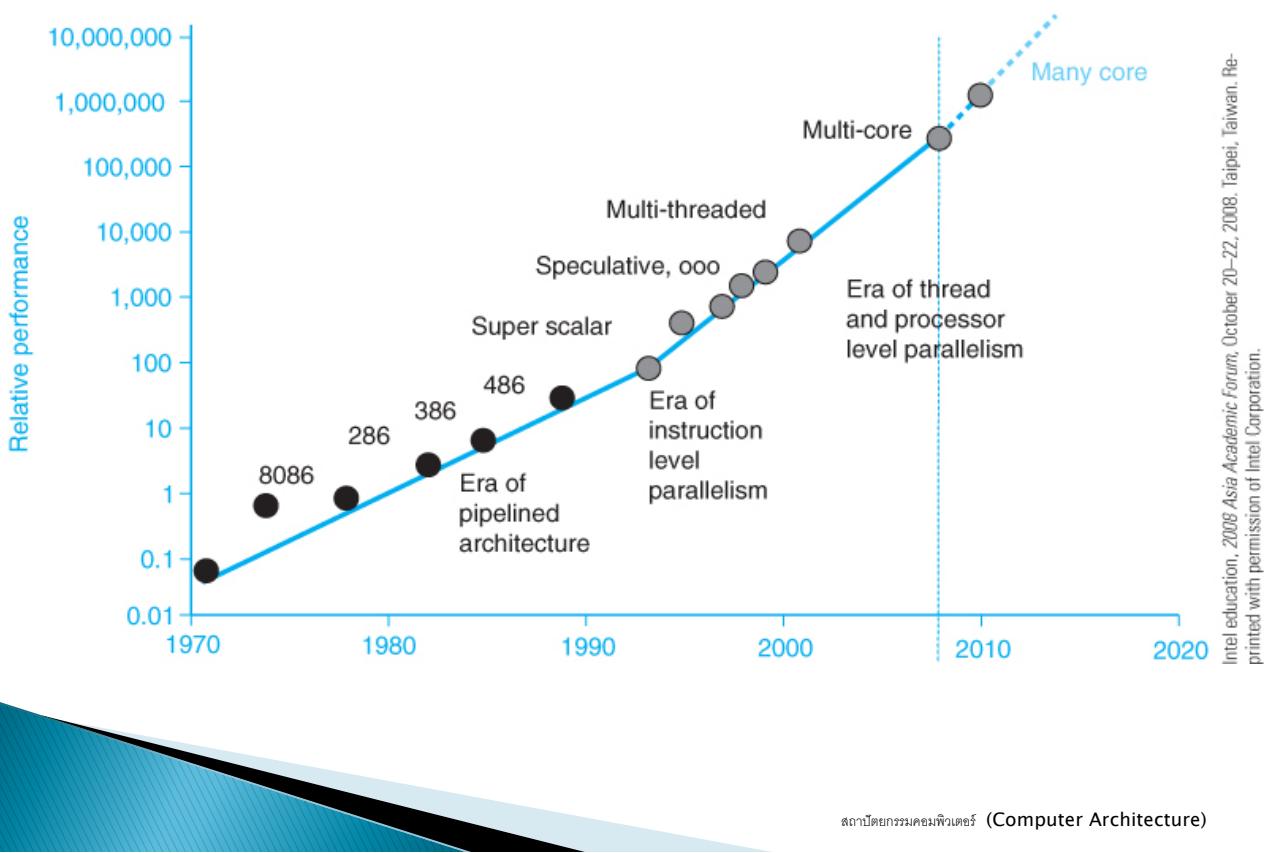
7

Growth of Processor Performance



© Cengage Learning 2014

Growth of Processor Performance



เครื่องคอมพิวเตอร์ชนิดมัลติ-thread (Multithread)

- ▶ ดำเนินงานหลาย-thread (Thread) พร้อมกันบนชาร์ดแวร์คนละชุด
 - โดยการมีรีจิสเตอร์ โปรแกรมเค้าท์เตอร์ (Program Counter) และ รีจิสเตอร์อื่นๆ หลายชุด
- ▶ มัลติ-threadชนิดละเอียด (Fine-grain multithreading)
 - เปลี่ยนthreadการทำงานทุกๆ ไซเคิลของโปรเซสเซอร์
 - ด้วยการสลับการทำงานทีละคำสั่งจากแต่ละthread (Interleave instruction execution)
 - หากคำสั่งใดค้าง เอาคำสั่งจากthreadอื่นมารันแทน
- ▶ มัลติ-threadชนิดหยาบ (Coarse-grain multithreading)
 - เปลี่ยนthreadการทำงานเมื่อเกิดการค้างนานๆ เช่น การค้นข้อมูลไม่เจอที่ L2 แคช เป็นต้น ทำให้ชาร์ดแวร์มีความซับซ้อนน้อยกว่าชนิดละเอียด

Multiple Processes Single Threaded vs. Multithreaded Processor (Single Core)

Multithreading improves pipeline efficiency.

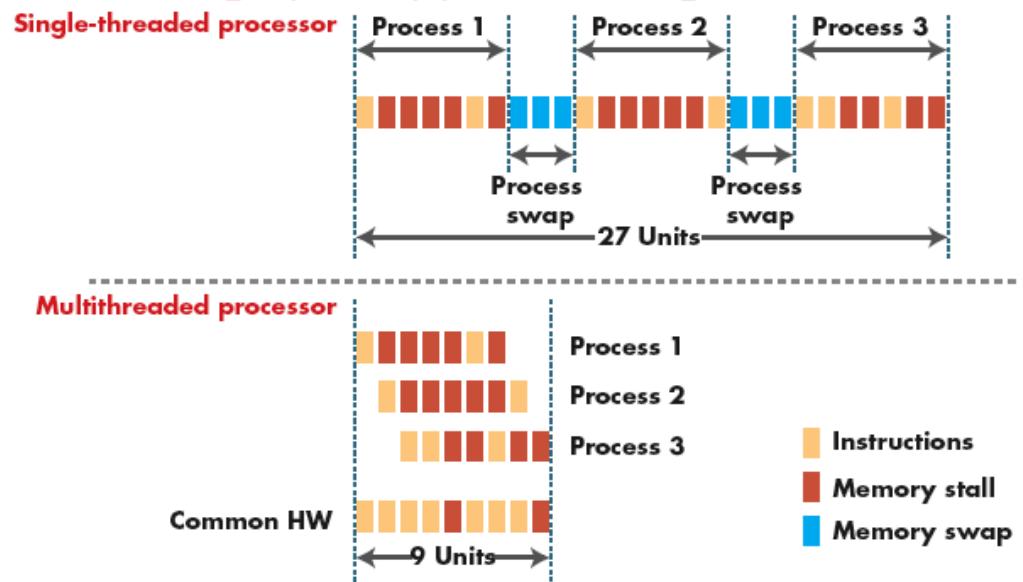


Figure 2

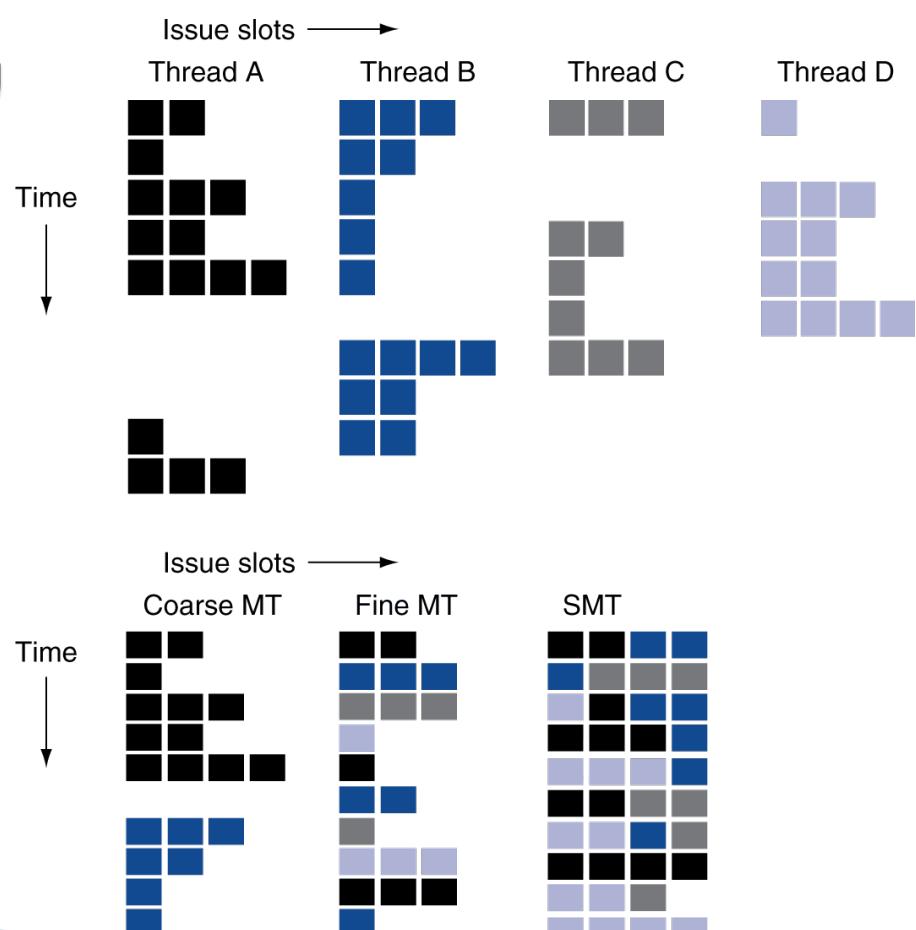
ສານເປົ້າຍກຮຽນຄວມພິຈາເຕຊ (Computer Architecture)

11

Multithreading Processes (Single Core)

- Coarse Grained
- Fine Grained
- Simultaneous

MultiThread
(SMT)



ສານເປົ້າຍກຮຽນຄວມພິຈາເຕຊ (Computer Architecture)

12

เครื่องคอมพิวเตอร์ชนิดมัลติ-thread (Multithread)

- ▶ กรณีโปรเซสเซอร์สามารถทำงานหลายคำสั่งพร้อมกันได้
 - สามารถทำงานคำสั่งจากหลายๆ เทรดพร้อมกันได้
 - คำสั่งจากเทรดที่ไม่เกี่ยวข้องกันสามารถทำงานพร้อมกันได้หาก hardware รองรับพร้อม
 - ภายในเทรดเดียวกัน การผูกโยงของคำสั่งจะต้องใช้การจัดลำดับ (Scheduling) และการเปลี่ยนชื่อรีจิสเตอร์ (Register renaming)
- ▶ ตัวอย่าง: Intel Pentium-4 HyperThread
 - รองรับได้ 2 เトレด: โดยการมีรีจิสเตอร์ 2 ชุด แต่ใช้งาน hardware และ cache ร่วมกัน
 - ใช้หลักการ SMT

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

13

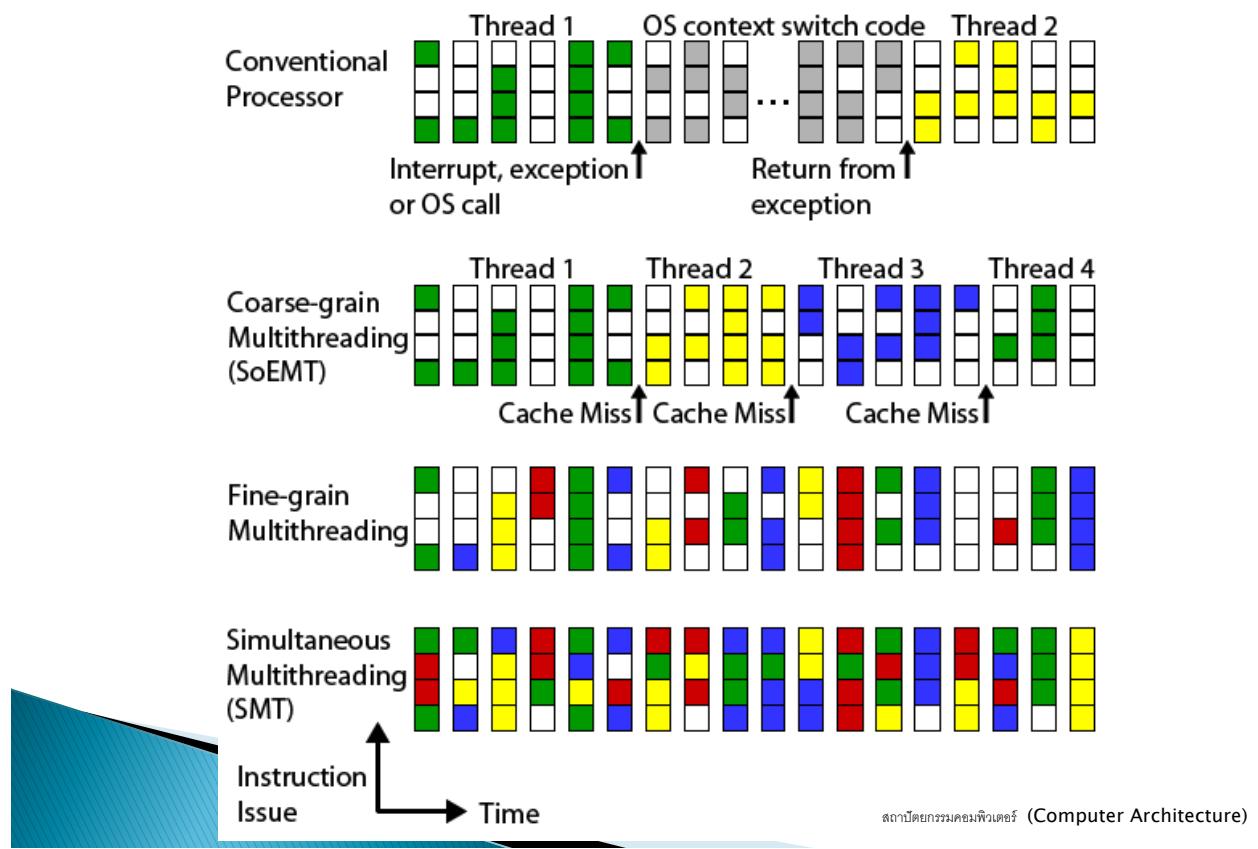
Multithreaded Process (Single Core)



สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

14

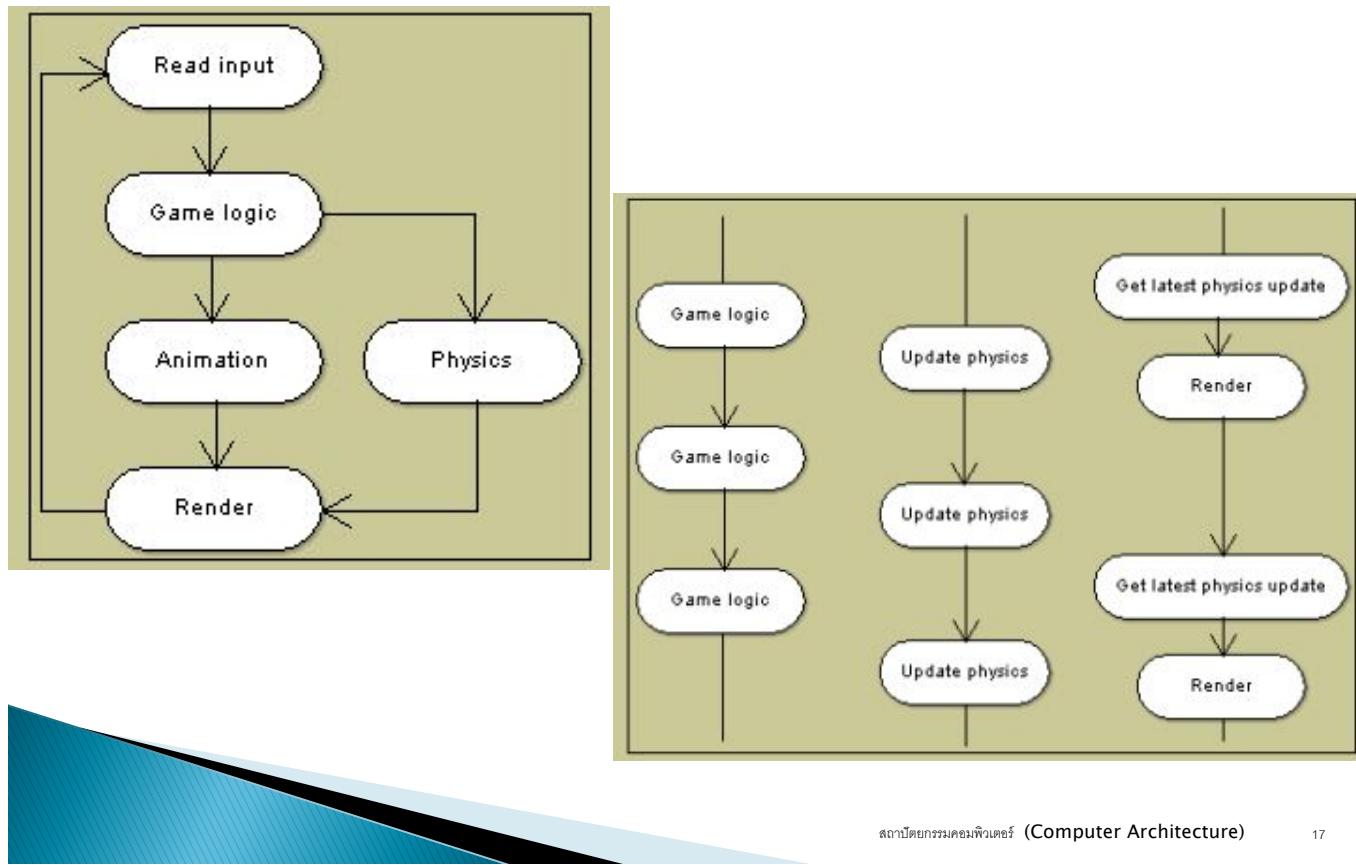
Multithreaded Process (Single Core)



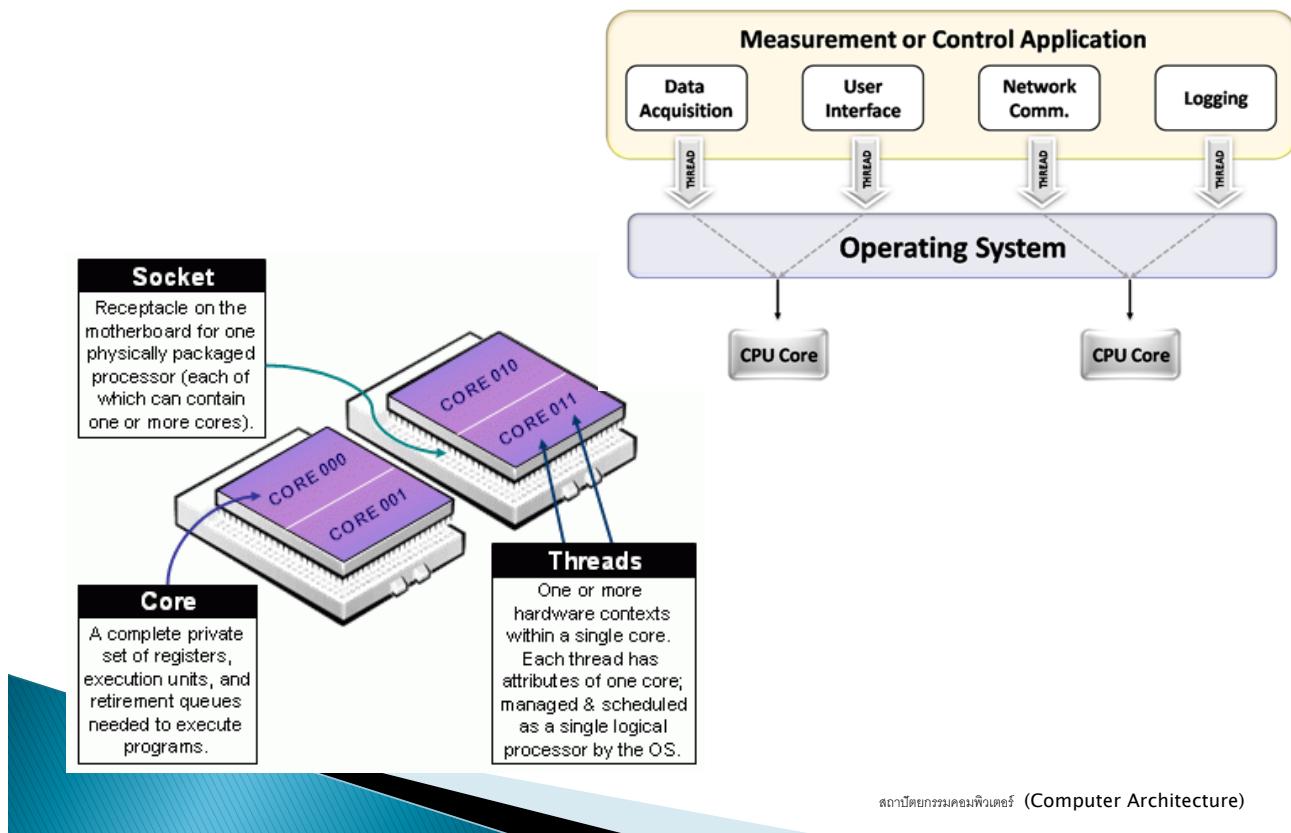
Multithreaded Programming

- ▶ มีความสำคัญมากขึ้น เนื่องจากอุปกรณ์คอมพิวเตอร์มีชีพิญจนิดมัลติคอร์
- ▶ ฟังก์ชันที่ถูกเรียกใช้บ่อยๆ สามารถเขียนหรือแปลงสภาพให้กลายเป็น Thread ของการทำงานบนชีพิญคอร์
- ▶ ระบบปฏิบัติการในปัจจุบันรองรับการทำงานแบบ Multithread
- ▶ ภาษาที่รองรับได้แก่ Java, C/C++ เป็นต้น

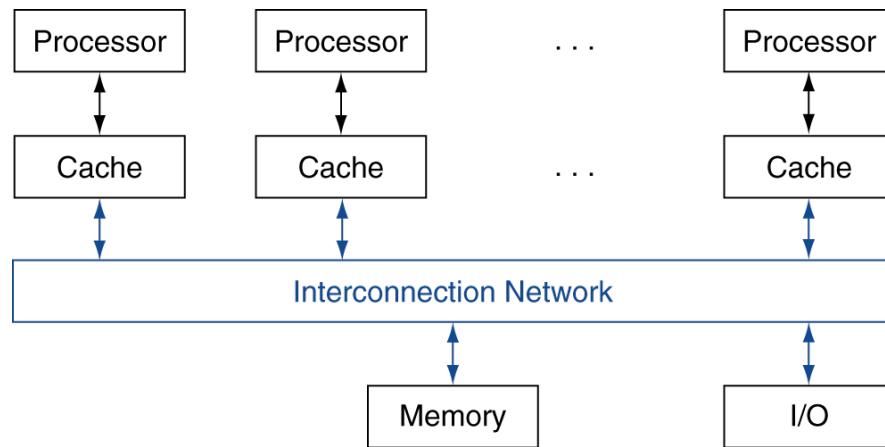
Case Study: 3D Game



Multithreaded Application (Multicore)



เครื่องคอมพิวเตอร์แบบขนาดชนิดใช้หน่วยความจำร่วมกัน (Shared Memory Multiprocessor: SMP) => Multicore



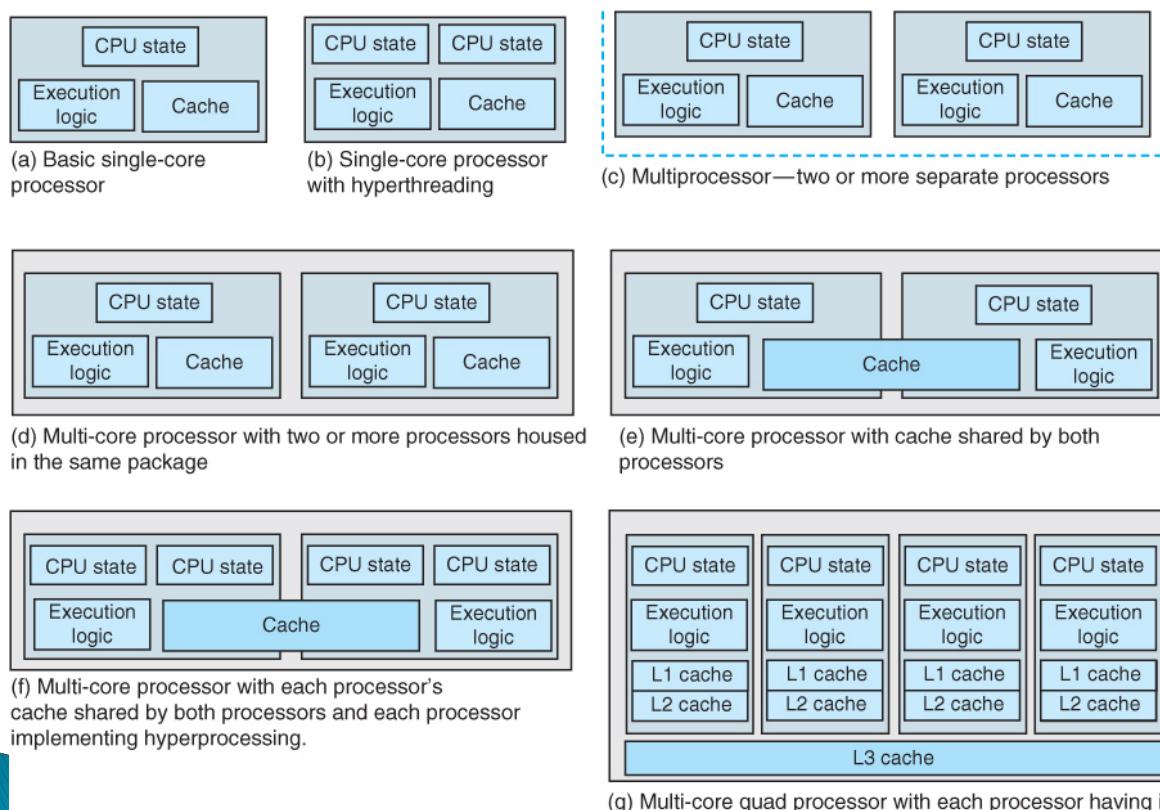
▶ ใช้หน่วยความจำหลักร่วมกัน (SMP: shared memory multiprocessor)

- การซิงโครไนเซชัน (Synchronization) ใช้ตัวแปรร่วม (shared variables) ต้องอาศัยการล็อกเวลาการเข้าถึงหน่วยความจำ (Memory access time): มีค่าคงที่ (Uniform Memory Access: UMA) vs. ไม่คงที่ (Non-Uniform Memory Access: NUMA)

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

19

Development of Multi-core Chips

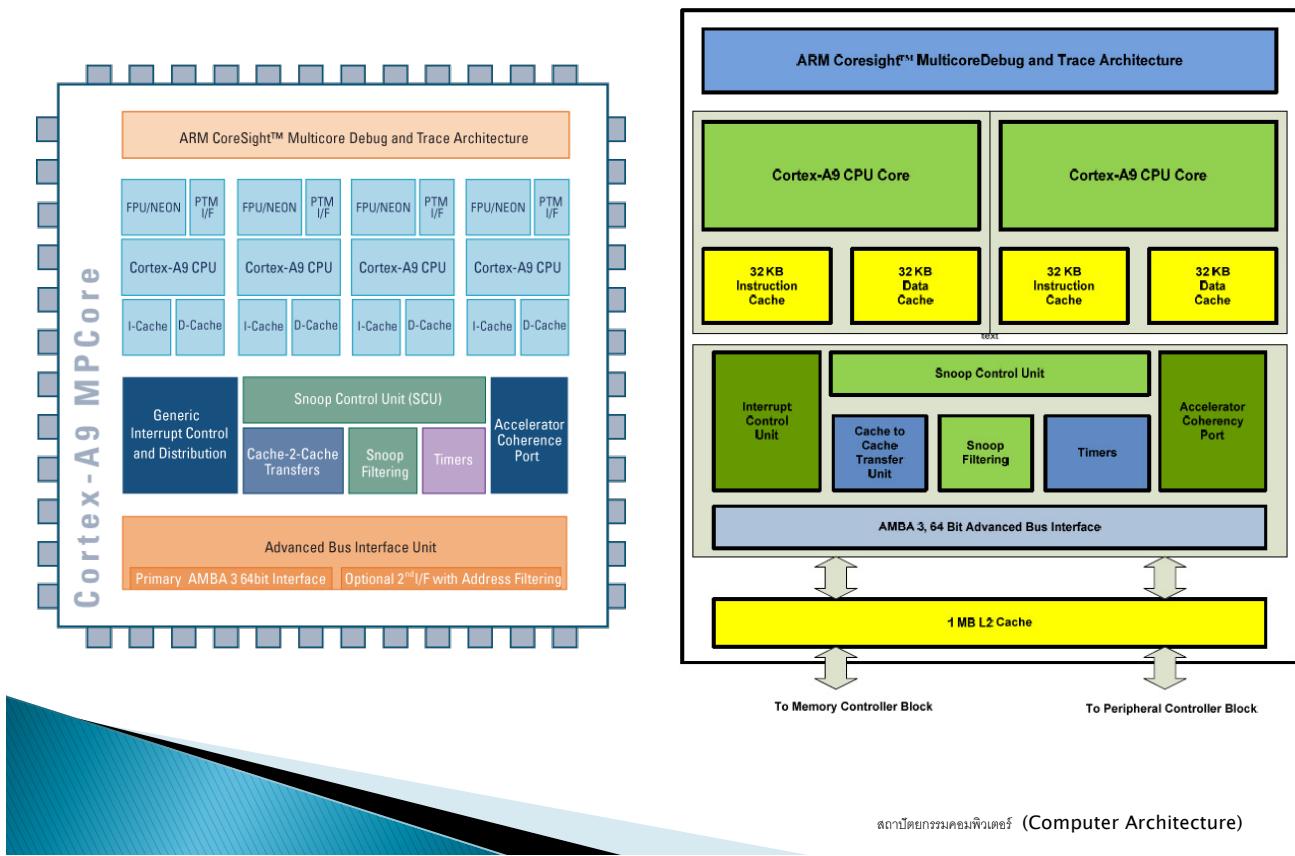


© Cengage Learning 2014

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

20

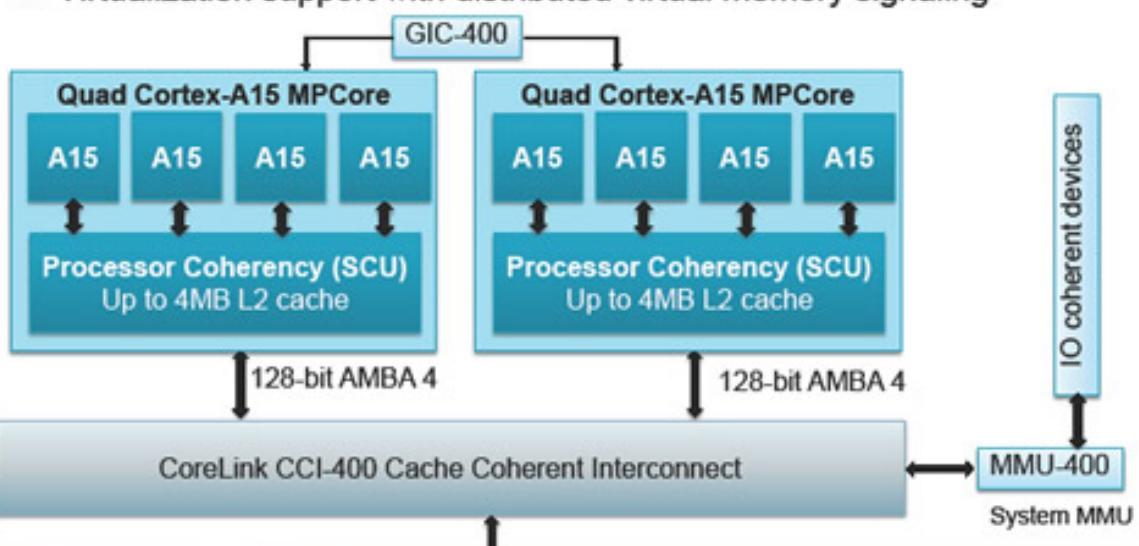
Multicore ARM



ARM Cortex A15 Scalability

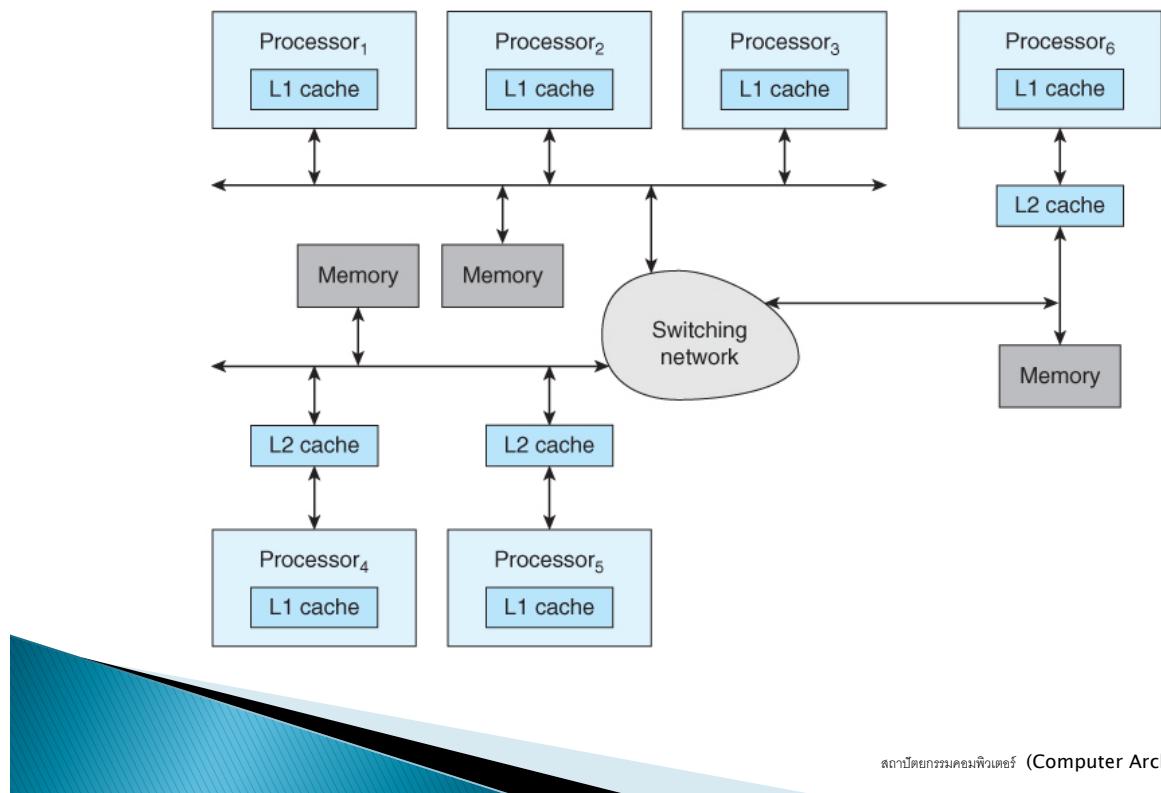
Introduces Cache Coherent Interconnect

- Processor to Processor Coherency and I/O coherency
- Memory and synchronization barriers
- Virtualization support with distributed virtual memory signaling



Non-Uniform Memory Access (NUMA)

FIGURE 13.14 A NUMA structure

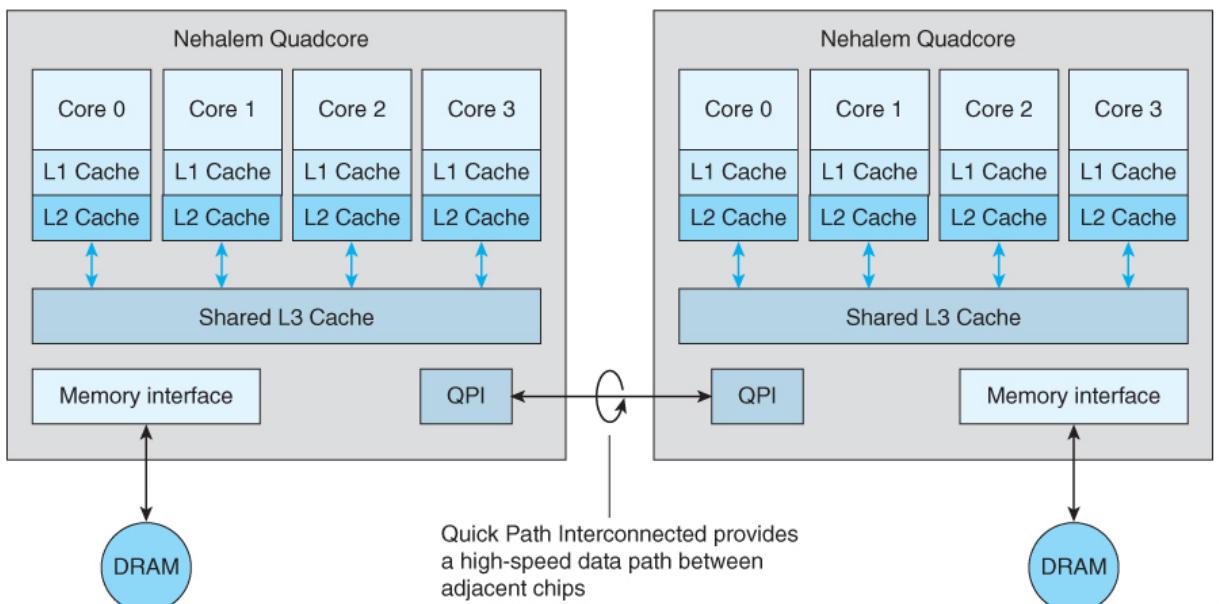


© Cengage Learning 2014

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

23

Intel Quick Path Interconnect (QPI)



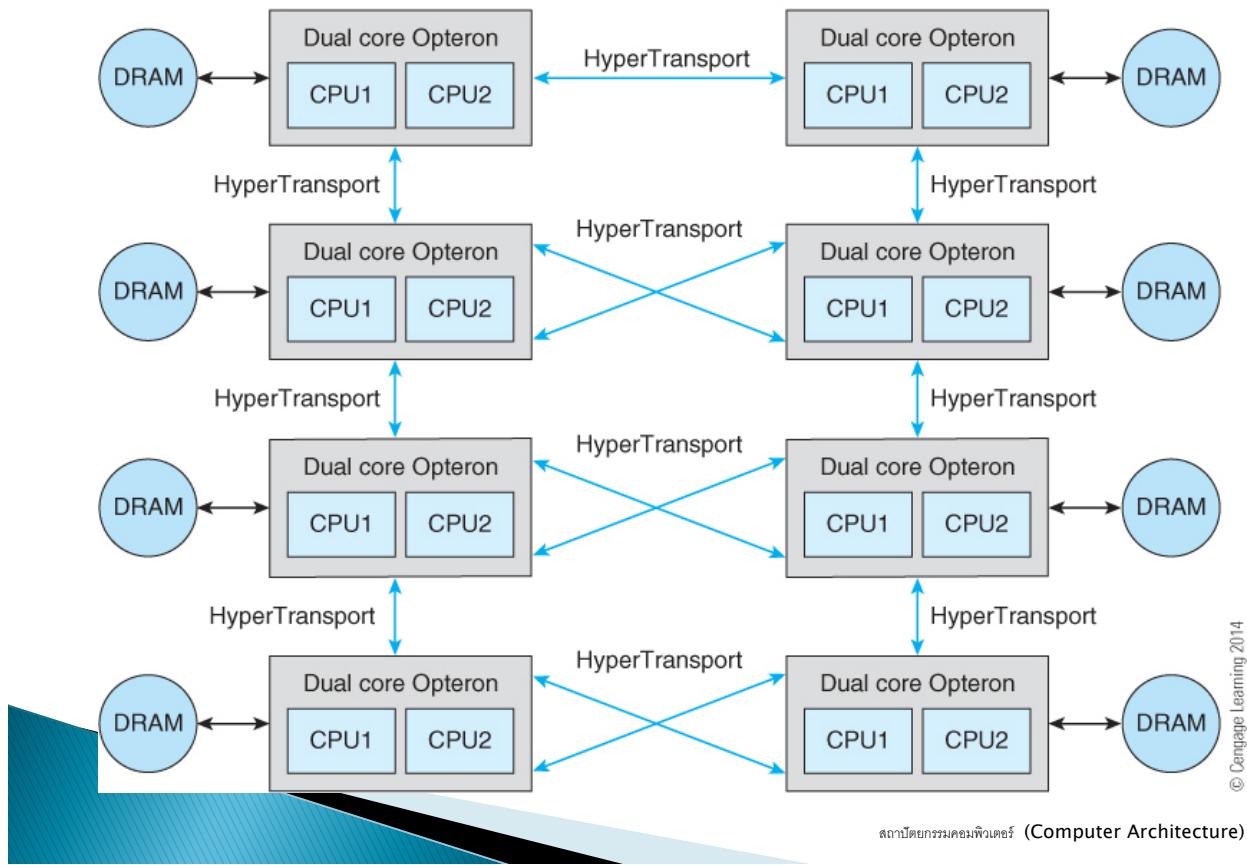
© Cengage Learning 2014

Quick Path Interconnected provides
a high-speed data path between
adjacent chips

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

24

AMD HyperTransport

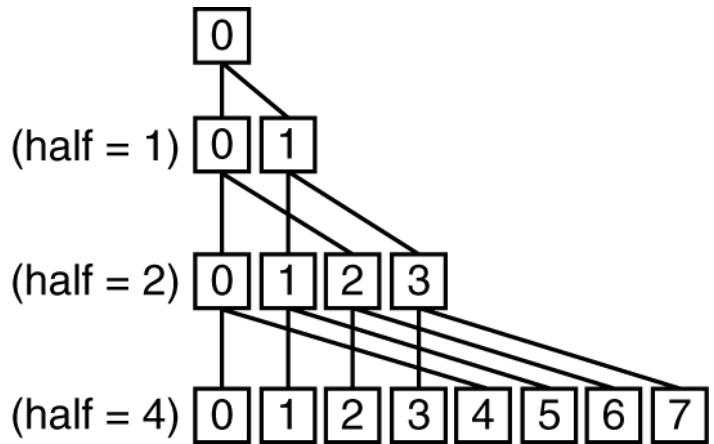


ตัวอย่าง: การหาผลบวกแบบขบวน

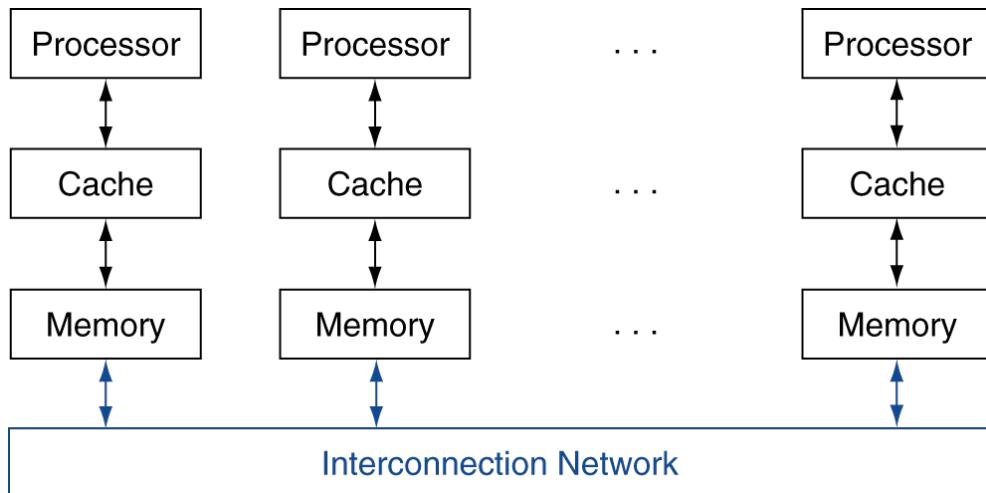
- ▶ จงบวกเลข 100,000 ตัวด้วยโปรเซสเซอร์ 100 ตัวชนิด UMA
 - แต่ละโปรเซสเซอร์มีเลข ID ประจำตัว คือ P_n โดย $0 \leq P_n \leq 99$
 - เริ่มจากการแบ่งตัวเลข 1000 ตัวต่อโปรเซสเซอร์ 1 ตัว
 - ดังนั้น ผลรวมบางส่วนบนโปรเซสเซอร์แต่ละตัวคำนวณได้ดังนี้
$$\text{sum}[P_n] = 0;$$
$$\text{for } (i = 1000*P_n; i < 1000*(P_n+1); i = i + 1)$$
$$\quad \text{sum}[P_n] = \text{sum}[P_n] + A[i];$$
- ▶ ทำการบวกค่าผลลัพธ์เข้าด้วยกัน ในแต่ละรอบจะต้องอาศัยหลักการกระจายการบวกให้แต่ละโปรเซสเซอร์ทำงานพร้อมกันให้มากที่สุด
 - ทำการรวมผลบวกแต่ละคู่ แล้วลดจำนวนการบวกลงครึ่งหนึ่งไปเรื่อยๆ
 - จะต้องมีการประสานงานผลบวกในแต่ละรอบ

ตัวอย่าง: การหาผลบวกแบบขนาน (ต่อ)

```
half = 100;  
repeat  
    synch();  
    if (half%2 != 0 && Pn == 0)  
        sum[0] = sum[0] + sum[half-1];  
        /* Conditional sum needed when half is odd;  
           Processor0 gets missing element */  
    half = half/2; /* dividing line on who sums */  
    if (Pn < half) sum[Pn] = sum[Pn] + sum[Pn+half];  
until (half == 1);
```



เครื่องคอมพิวเตอร์แบบขนานชนิดหน่วยความจำอิสระจากกัน



- ▶ มัลติคอมพิวเตอร์ (Multi Computer)
- ▶ แต่ละโปรเซสเซอร์เห็นหน่วยความจำแยกอิสระจากกัน
- ▶ การสื่อสารระหว่างกันใช้การส่งข้อความระหว่างกัน (Message Passing)

เครื่องคอมพิวเตอร์แบบขนาดชนิดหน่วยความจำอิสระจากกัน

- ▶ เครือข่ายของเครื่องคอมพิวเตอร์ที่อิสระจากกัน
- ▶ เหมาะกับโปรแกรมที่ทำงานร่วมกันแบบห่าง (Loosely Coupled)
 - เว็บเซิร์ฟเวอร์ ฐานข้อมูล การจำลองการทำงาน (Simulation)
- ▶ เพิ่ม High availability, เพิ่ม Scalability, และราคาไม่แพง
- ▶ อุปสรรคและปัญหา
 - ต้นทุนการบริหารจัดการ
 - การเชื่อมต่อระหว่างเครื่องมีแบบดีวิดธ์ไม่สูงมาก (ระดับ 1-10 จิกะบิทต่อวินาที)
 - แบนด์วิดธ์ระหว่าง processor/memory บนเครื่อง SMP สูงกว่ามาก (ระดับ 10 จิกะบิทต่อวินาทีขึ้นไป)

เครื่องคอมพิวเตอร์แบบขนาดชนิดหน่วยความจำอิสระจากกัน

- ▶ กริดคอมพิวติ้ง (Grid Computing)
 - เครื่องคอมพิวเตอร์ที่กระจายอยู่ทั่วโลกและเชื่อมต่อกับเครือข่ายอินเทอร์เน็ท
 - กระจายงานออกไปยังเครื่องต่างๆ เมื่อคำนวณเสร็จแล้วจึงส่งผลกลับ
 - โปรแกรมจะทำงานโดยอัตโนมัติ ขณะที่ผู้ใช้ปล่อยให้เครื่องอยู่เฉยๆ (Idle)
 - ตัวอย่าง โครงการ SETI@home, World Community Grid

ตัวอย่าง: การหารผลบวก บนเครื่องแบบ Message Passing

- ▶ จงคำนวณหารผลบวกของตัวเลขจำนวน 100,000 ตัวด้วยโปรเซสเซอร์ 100 ตัว
- ▶ หากผลบวกของตัวเลขจำนวน 100,000 ตัวด้วยเครื่องคอมพิวเตอร์ 100 เครื่อง
 - แต่ละเครื่องมีเลข ID ประจำตัว คือ P_n โดย $0 \leq P_n \leq 99$
 - กระจายตัวเลข 1000 ตัวต่อคอมพิวเตอร์ 1 เครื่อง
 - ดังนั้น ผู้รวมบางส่วนในแต่ละเครื่องสามารถคำนวณได้ดังนี้
 - $sum = 0;$

```
for (i = 0; i < 1000; i = i + 1)
    sum = sum + AN[i];
```
- ▶ ทำการบวกค่า sum ของแต่ละเครื่องเข้าด้วยกัน ในแต่ละรอบจะต้องอาศัยหลักการ
 - กระจายการบวกให้แต่ละโปรเซสเซอร์ทำงานพร้อมกันให้มากที่สุด
 - เครื่องที่มีเลข ID มากส่งผลบวกมาให้เครื่องที่มีเลข ID น้อย เพื่อทำการบวก
 - จำนวนเครื่องที่บวกจะลดจำนวนลงเหลือครึ่งหนึ่งไปเรื่อยๆ

ตัวอย่าง: การหารผลบวก บนเครื่องแบบ Message Passing

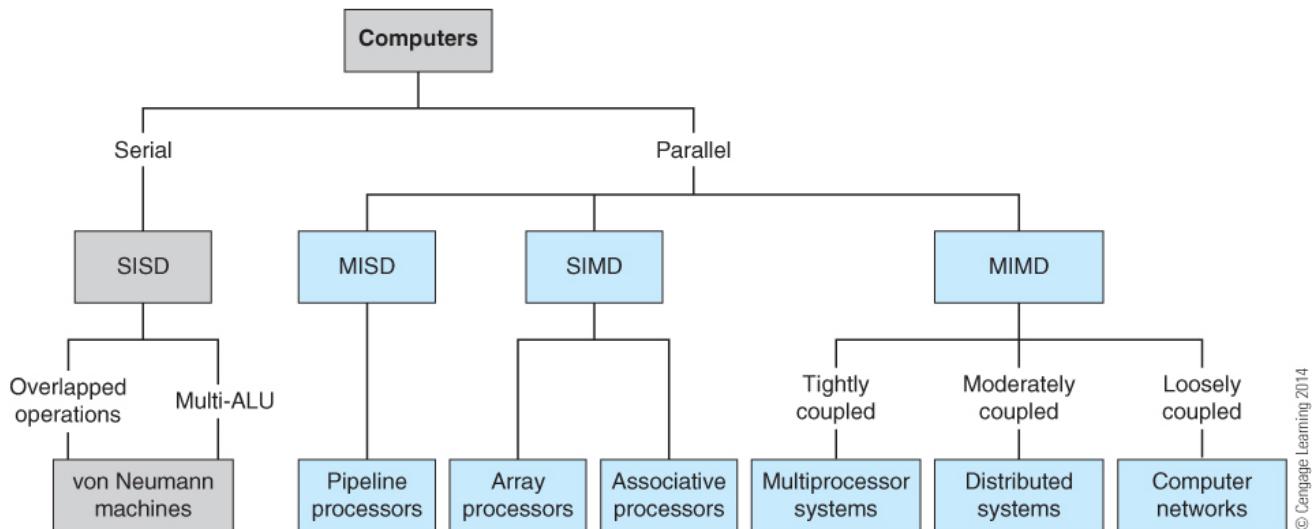
- ▶ กำหนดให้ใช้ฟังก์ชัน `send()` and `receive()`

```
limit = 100; half = 100; /* 100 processors */
repeat
    half = (half+1)/2;      /* send vs. receive dividing line */
    if (Pn >= half && Pn < limit)
        send(Pn - half, sum);
    if (Pn < (limit/2))
        sum = sum + receive();
    limit = half;           /* upper limit of senders */
until (half == 1); /* exit with final sum */
```

- สมมติว่า `send/receive` ใช้เวลาทำงานนานพอๆ กับการบวกเลข
- การใช้ `Send/receive` ทำให้เกิดการ synchronization

อนุกรมวิธานของ Flynn (Flynn's Taxonomy)

FIGURE 13.4 Computer classifications—the Flynn view



© Cengage Learning 2014

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

33

สตรีมคำสั่งและข้อมูล (Instruction and Data Streams)

▶ ใช้หลักการกายวิภาคของ Flynn (Flynn's Taxonomy)

		Data Streams (สตรีมข้อมูล)	
		เดี่ยว (Single)	หลาย (Multiple)
สตรีมคำสั่ง (Instruction Streams)	เดี่ยว (Single)	SISD: Intel Pentium 4	SIMD: SSE instructions of x86
	หลาย (Multiple)	MISD: No examples today	MIMD: Intel Xeon e5345

■ การทำงานชนิด SPMD (Single Program Multiple Data)

- ▶ โปรแกรมเดียวกันทำงานแบบขนานบนเครื่องคอมพิวเตอร์ชนิด MIMD
- ▶ มีเงื่อนไขในการทำงานบนโปรเซสเซอร์แต่ละตัว (คอร์)

การคำนวณแบบขنانอื่นๆ

- ▶ ใช้จีพียู (GPU: Graphic Processing Unit) บนการ์ดจอ
- ▶ การประมวลสามารถทำแบบขนานได้เนื่องจากข้อมูลมี Data Parallelism สูง
 - GPUs มีลักษณะการทำงานแบบ multithreaded
 - ใช้การสวิทช์เทรด (Thread switching) เพื่อช่อนเวลาในการอ้างถึงหน่วยความจำ
 - ทำให้ไม่จำเป็นต้องพึ่งพาแคชหลายระดับมากนัก
 - หน่วยความจำด้านกราฟิกสมีความกว้าง (Width) มากทำให้แบบดิจิตอลสูง
- ▶ มีแนวโน้มที่จะพัฒนาให้เป็น GPU สารพัดประโยชน์ (general purpose)
 - เพื่อช่วย CPU ในการคำนวณ
 - โดย CPU เหมาะกับงานแบบอนุกรม, GPU เหมาะกับงานแบบขนาน
- ▶ ภาษาที่ใช้ในการพัฒนาโปรแกรม (Programming languages/APIs)
 - DirectX ของไมโครซอฟต์, OpenGL, OpenCL
 - C for Graphics (Cg), High Level Shader Language (HLSL)
 - Compute Unified Device Architecture (CUDA) ของ nVidia

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

35

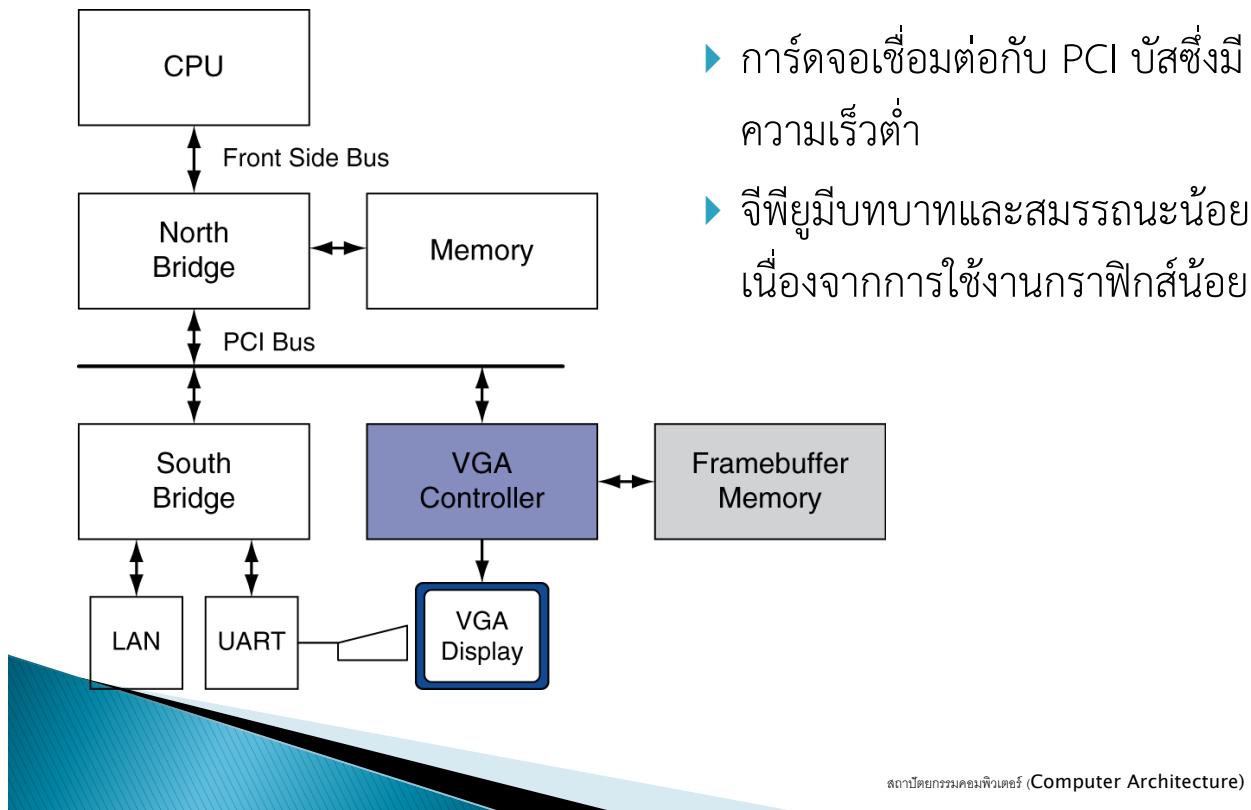
วิวัฒนาการของการ์ดจอ

- ▶ การ์ดจอ (VGA Card) ในยุคแรกๆ
 - ใช้หน่วยความจำเก็บค่าแต่ละจุดและการสร้างแอดเดรสในการสร้างสัญญาณภาพบนจอ
- ▶ การ์ดจอยุคที่สามารถประมวลผลกราฟิกส์ 3 มิติ
 - เดิมมิใช่ในคอมพิวเตอร์ที่ราคาแพง
 - จากกฎของ Moore's Law ทำให้ราคาลดลงและความสามารถเพิ่มขึ้น
 - ปัจจุบันการ์ดจอ 3 มิติมิใช่ในเครื่องคอมพิวเตอร์ทั่วไปและเครื่องเล่นเกมส์
- ▶ จีพียู (Graphics Processing Units)
 - เป็นโปรเซสเซอร์ที่ใช้งานด้าน 3 มิติโดยตรง
 - มีพังก์ชันการประมวลผลข้อมูล Vertex/pixel, พังก์ชันการทำ shading, การแมป texture mapping เป็นต้น

สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

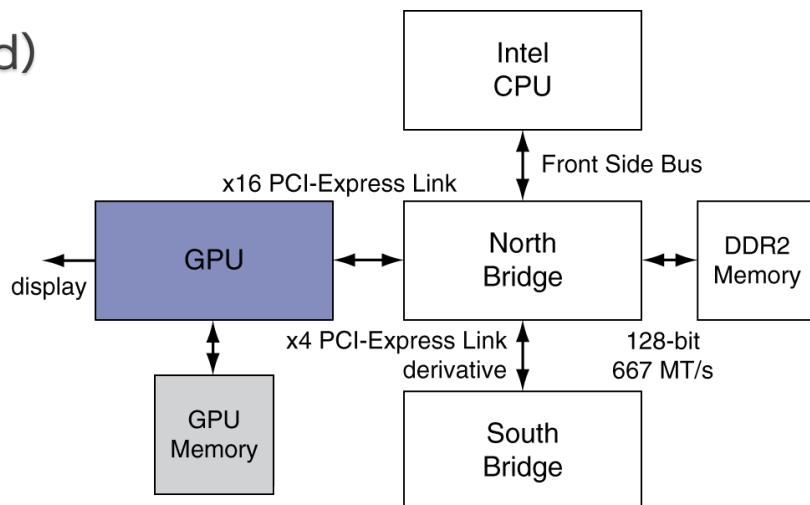
36

การ์ดจอ (VGA Card) ในยุคแรกๆ



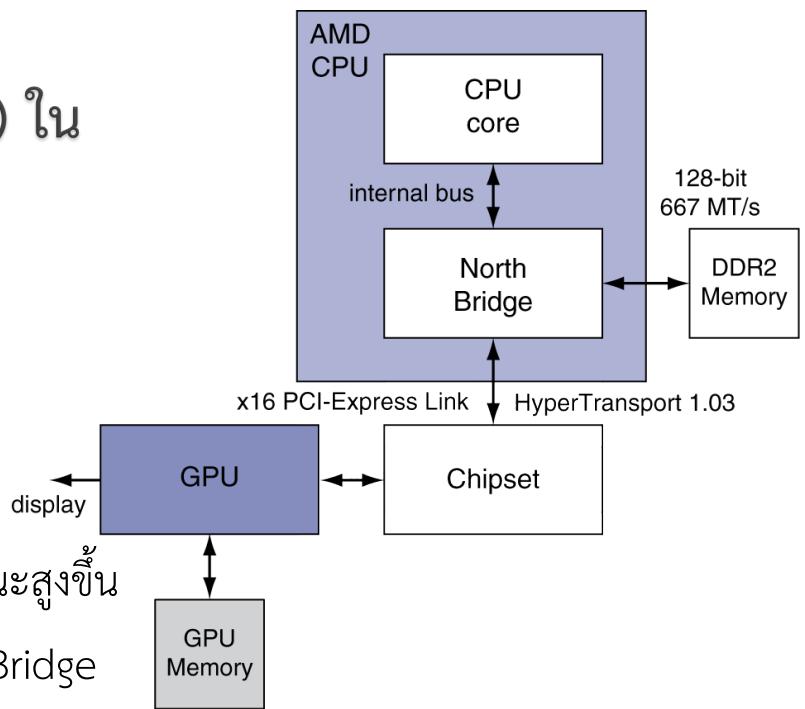
การ์ดจอ (VGA Card)

ในยุคกลาง



- ▶ ความเร็วบัสเพิ่มขึ้นเป็น PCI Express
- ▶ จีพียูมีบทบาทและสมรรถนะสูงขึ้น เนื่องจากการใช้งานกราฟิกเพิ่มสูงขึ้น
- ▶ ปรับตำแหน่งของจีพียูให้เข้าใกล้ชีพียูมากขึ้น
- ▶ หน่วยความจำของ GPU มีความจุเพิ่มมากขึ้นและความเร็วสูงขึ้น

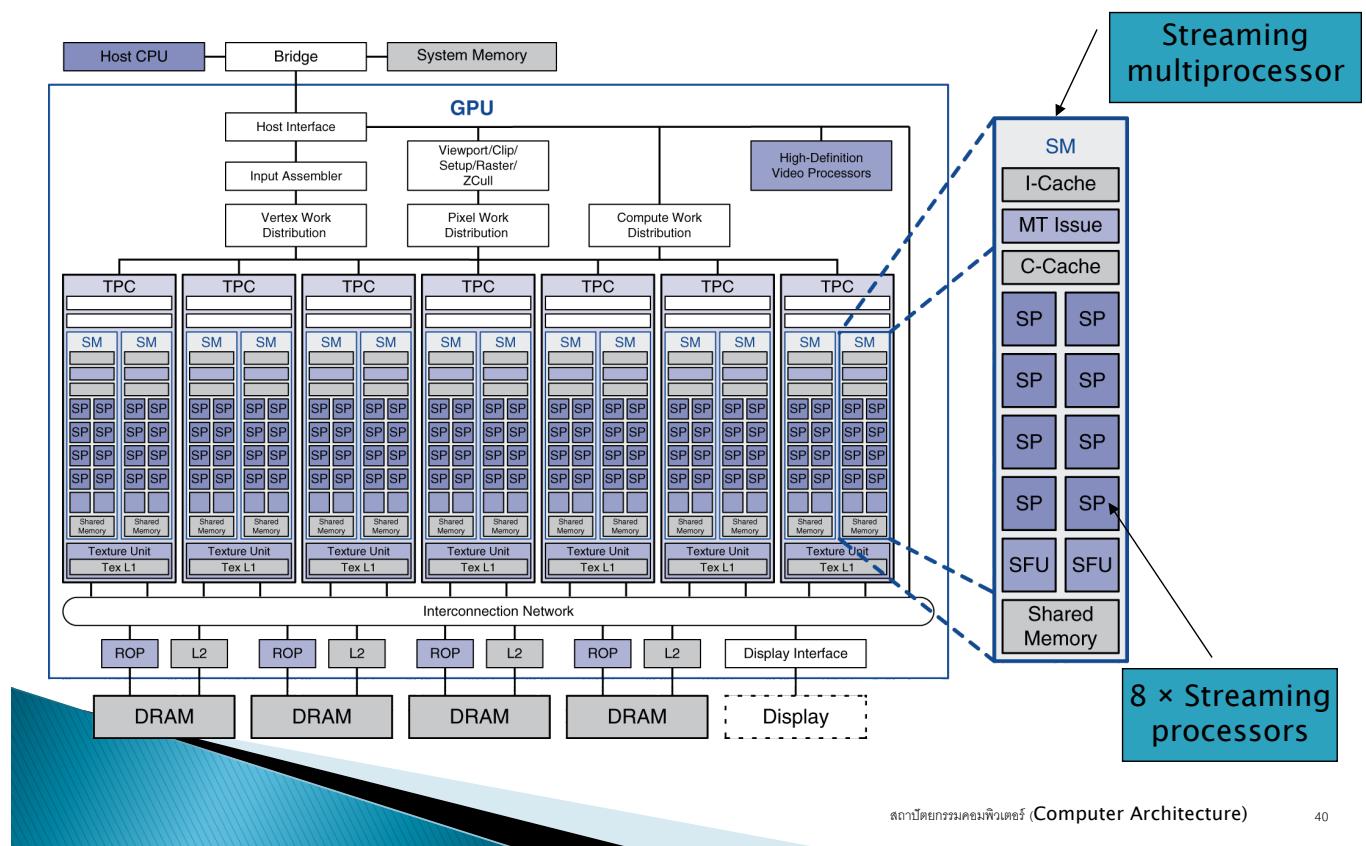
การดจօ (VGA Card) ใน ယុគកែនប័ត្រុប៉ា



ភាពវិទ្យកម្មករណី (Computer Architecture)

39

តัวอย่าง: ជីពិឃួរនៃ NVIDIA ឱៗ Tesla



ភាពវិទ្យកម្មករណី (Computer Architecture)

40

การจำแนกจีพียูด้านการคำนวณแบบขنان

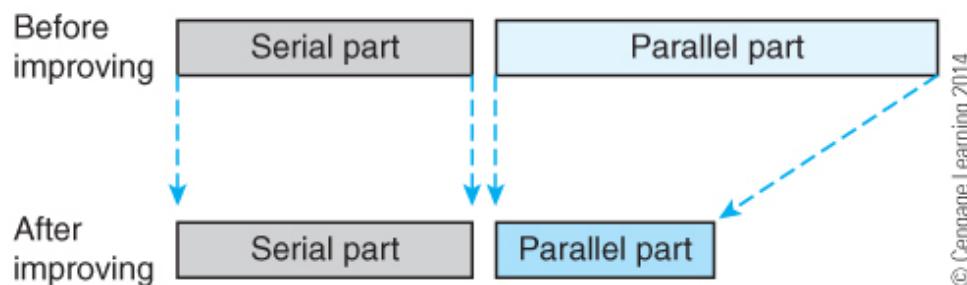
Parallelism ระดับต่างๆ	ค้นพบ ณ เวลาคอมไพล์ชอร์สโค๊ด (Discovered at Compile Time)	ค้นพบ ณ เวลารันโปรแกรม (Discovered at Runtime)
ระดับคำสั่ง (Instruction-Level Parallelism)	VLIW (Very Long Instruction Word)	Superscalar
ระดับข้อมูล (Data-Level Parallelism)	SIMD or Vector	Tesla Multiprocessor



สถาปัตยกรรมคอมพิวเตอร์ (Computer Architecture)

41

Amdahl's Law



© Cengage Learning 2014

- ▶ หลังจากที่เปลี่ยนการทำงานของ Parallel part ให้ทำงานแบบขนานสำเร็จ
 - ▶ ระยะเวลาการทำงานอาจลดลง
 - ▶ สามารถคำนวณค่า Speedup (S) ได้ดังต่อไปนี้



ສາງປົກຍາວຂະໜາດອານຸພິວເຕີຣີ (Computer Architecture)

10

Total Speedup (S)

$$S = \frac{1}{f_s + \frac{1-f_s}{p}} = \frac{p}{pf_s + 1 - f_s}$$

f_s = serial fraction

f_p = parallel fraction

$f_s + f_p = 1$

p = number of parallel processors

Total Speedup (S)

TABLE 6.9 Effect of Amdahl's Law

Processors	Speedup Ratio $f_s = 0.2$	Speedup Ratio $f_s = 0.1$
1	1	1
2	1.667	1.818
3	2.143	2.500
4	2.500	3.077
5	2.778	3.571
10	3.571	5.263
100	4.808	9.174
∞	5.000	10.00

- ▶ เมื่อมีจำนวนคอร์/processor เพิ่มขึ้น สามารถเพิ่มความเร็วได้
- ▶ เมื่อ f_s ลดลง ความเร็ว S มีโอกาสเพิ่มสูงขึ้นมากกว่า

สรุป

- ▶ การคำนวณแบบอนุกรมและแบบขนาน ทำให้วิวัฒนาการของคอมพิวเตอร์ในมิติเวลาทางคอมพิวเตอร์แบบขนานมากขึ้น
- ▶ คอมพิวเตอร์แบบขนานมีหลายชนิด ได้แก่ ชนิดหน่วยความจำอิสระจากกัน ชนิดหน่วยความจำร่วมกัน
- ▶ การพัฒนาซอฟต์แวร์แบบขนาน ขึ้นอยู่กับปัญหาที่มีการขนาดของข้อมูล หรือขนาดของการทำงาน
- ▶ ภายวิภาคของ Flynn (Flynn's Taxonomy) สามารถนำไปใช้เคราะห์การทำงานของตัวประมวลผลแบบต่างๆ ได้
- ▶ กฎของ Amdahl สามารถอธิบาย Speedup ของการคำนวณแบบขนาน

คำถามท้ายบท

- ▶ จะวิเคราะห์การ sort ชนิดต่างๆ เหล่านี้ เช่น Quick Sort และ Bubble Sort ว่ามี Parallelism ชนิดใดบ้าง
- ▶ จะวิเคราะห์ การคำนวณแบบ Super Pipeline/Super Scalar ในชีพียูปัจจุบัน ตามภายวิภาคของ Flynn
- ▶ จะวิเคราะห์ ตัวอย่างการคำนวณหาผลลบวกแบบขนานตามภายวิภาคของ Flynn