Chương 11: Kỹ thuật ống dẫn và kiến trúc nâng cao

TS. Phạm Công Thắng
Bộ môn hệ thống nhúng
Khoa Công Nghệ Thông Tin
Trường Đại học Bách Khoa
Đại học Đà Nẵng

- Kiến trúc RISC và CISC
- Pipeline
- Superscalar Processors
- Branch Prediction, Register Renaming
- Multiprocessor
- Mô hình SIMD
- Mô hình MIMD
- Mạng kết nối

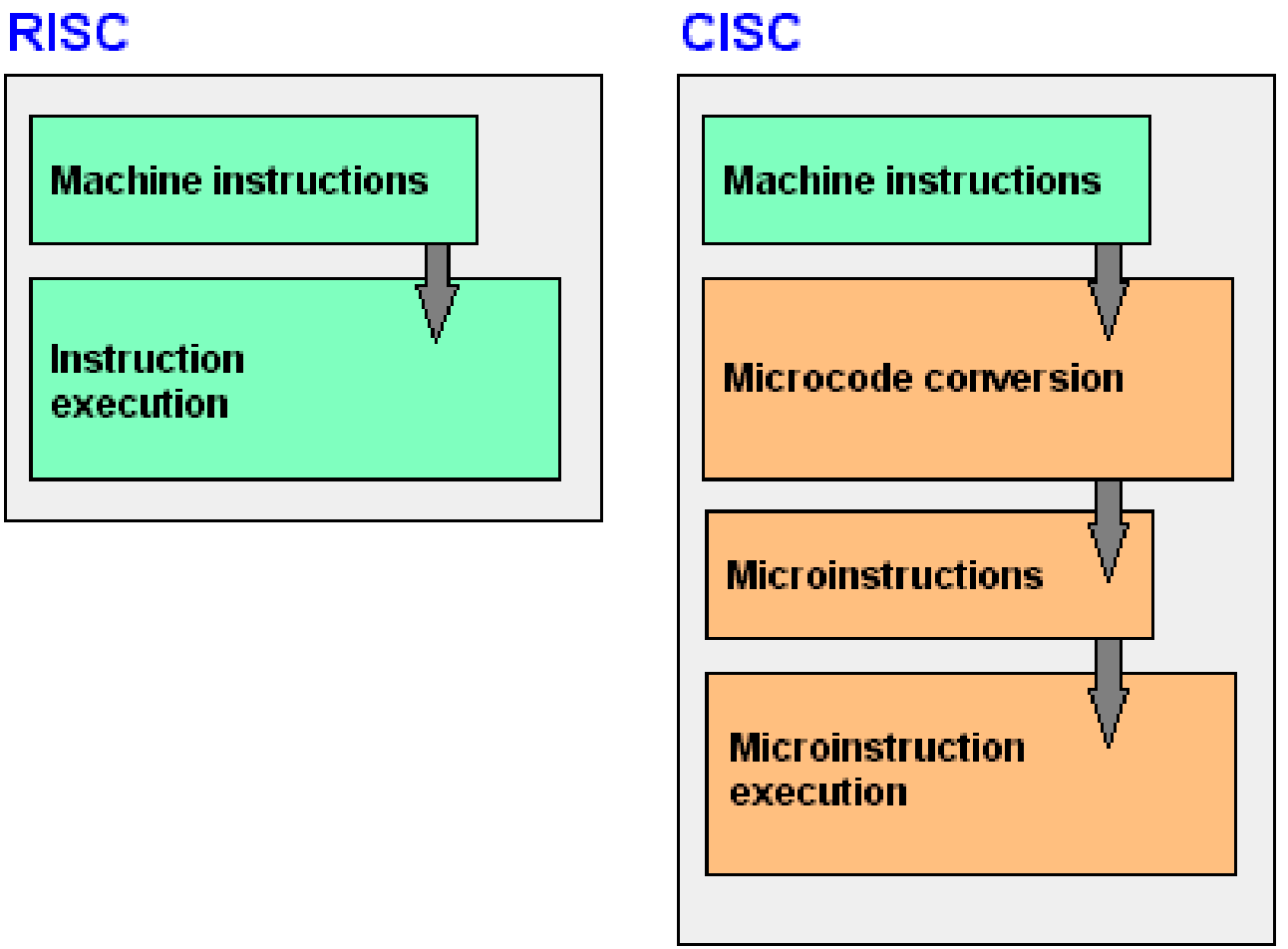
Kiến trúc CISC và RISC

- CISC (Complex Instruction Set Computer)
  - Máy tính với tập lệnh phức tạp, chẳng hạn như dòng chip x86 của Intel).
  - Những nhà thiết kế VXL cố gắng để mỗi lệnh có thể thực hiện càng nhiều chức năng càng tốt. Điều này dẫn đến một lệnh sẽ làm tất cả công việc
    - Ví dụ nạp 2 số cần cộng, cộng chúng lại, và cuối cùng lưu trở lại vào bộ nhớ. Cũng lệnh đó lại có thể đọc một số từ thanh ghi và số còn lại từ bộ nhớ sau đó lưu kết quả vào bộ nhớ.

Kiến trúc CISC và RISC

- RISC (viết tắt của Reduced Instructions Set Computer)
  - Máy tính với tập lệnh đơn giản hóamột phương pháp thiết kế các bộ vi xử lý (VXL) theo hướng đơn giản hóa tập lệnh, trong đó thời gian thực thi tất cả các lênh đều như nhau.
  - Hiện nay các bộ vi xử lý RISC phổ biến là ARM, SuperH, MIPS, SPARC, DEC Alpha, PA-RISC, PIC, và PowerPC của IBM.

Kiến trúc CISC và RISC



Kiến trúc CISC và RISC

- Khác biệt giữa RISC so với CISC
  - CISC
    - Được thiết kế nhằm tạo thuận lợi cho các nhà lập trình ứng dụng bằng cách rút gọn nhiều câu lệnh đơn giản, thông dụng thành một câu lệnh thực thi dài. Điều này làm cho CISC xử lý chậm hơn nhưng lại đạt yếu tố thân thiện.
  - RISC
    - Thực hiện nhanh nhưng kém thân thiện hơn, mỗi câu lệnh đơn giản trong RISC phục vụ cho một mục đích hẹp rất cụ thể, thực hiện rất nhanh và các lệnh này được tiến hành song song. RISC đòi hỏi nhà lập trình phải kiên nhẫn, giỏi và một trình biên dịch được tối ưu kỹ lưỡng.

Pipeline

- Trong các máy tính hiện đại, CPU được tổ chức để song song hoá nhiều công đoạn trong một chu kỳ xử lý lệnh.
- CPU không chỉ lấy từng lệnh ở bộ nhớ mà lấy cả khối lệnh đặt sẵn trên cache để giảm thiểu thời gian do truy cập bộ nhớ nhiều lần.
- Khi nhiều lệnh đã được đưa lên cache thì trong khi đang thực hiện một lệnh, có thể đồng thời đọc dữ liệu cho một lệnh thứ hai và giải mã một lệnh thứ 3 theo thứ tự. Cơ chế này gọi là pipeline (đường ống)

Pipeline

- Giả sử có 5 lệnh và mỗi lệnh được thực hiện trong cùng một khoản thời gian. Mỗi lệnh được thực hiện trong 5 giai đoạn va mỗi giai đoạn được thực hiện trong 1 chu kỳ xung nhịp.
- Các giai đoạn thực hiện một lệnh là:
  - Lấy lệnh (IF:Instruction Fetch),
  - Giải mã (ID: Instruction Decode),
  - Thi hành (EX: Execute),
  - Truy nhập bộ nhớ (MEM: Memory Access),
  - Lưu trữ kết quả (RS: Result Storing).

Pipeline

- Kiểu xử lý tuần tự thong thường, 5 lệnh được thực hiện trong 25 chu kỳ xung nhịp
- Thi xử ly lệnh theo kỹ thuật ống dẫn thực hiện 5 lệnnh chỉ trong 9 chu kỳ xung nhịp.

Chuỗi lệnh	Chu kỳ xung nhịp								
	1	2	3	4	5	6	7	8	9
Lệnh thứ i	IF	ID	EX	MEM	RS				
Lệnh thứ i+1		IF	ID	EX	MEM	RS			
Lệnh thứ i+2			IF	ID	EX	MEM	RS		
Lệnh thứ i+3				IF	ID	EX	MEM	RS	
Lệnh thứ i+4					IF	ID	EX	MEM	RS

Pipeline

- Kỹ thuật ống dẫn làm tăng tốc độ thực hiện các lệnh. Tuy nhiên kỹ thuật ống dẫn có một số ràng buộc:
  - Cần phải có một mạch điện để thi hành mỗi giai đoạn của lệnh vì tất cả các giai đoạn của lệnh được thi hành cùng lúc.
    - Trong một bộ xử lý không dùng kỹ thuật ống dẫn, ta có thể dùng bộ làm toán ALU để cập nhật thanh ghi PC, cập nhật địa chỉ của toán hạng bộ nhớ, địa chỉ ô nhớ mà chương trình cần nhảy tới, làm các phép tính trên các toán hạng vì các phép tính này có thể xảy ra ở nhiều giai đoạn khác nhau.
  - Phải có nhiều thanh ghi khác nhau dùng cho các tác vụ đọc và viết (trong ví dụ tại một chu kỳ xung nhịp, ta thấy cùng một lúc có 2 tác vụ đọc (ID, MEM) và 1 tác vụ viết (RS).

Pipeline

- Kỹ thuật ống dẫn có một số ràng buộc (tiếp)
  - Trong một máy có kỹ thuật ống dẫn, có khi kết quả của một tác vụ trước đó, là toán hạng nguồn của một tác vụ khác. Như vậy sẽ có thêm những khó khăn mà ta sẽ đề cập ở mục tới.
  - Cần phải giải mã các lệnh một cách đơn giản để có thể giải mã và đọc các toán hạng trong một chu kỳ duy nhất của xung nhịp.
  - Cần phải có các bộ làm tính ALU hữu hiệu để có thể thi hành lệnh số học dài nhất, có số giữ, trong một khoảng thời gian ít hơn một chu kỳ của xung nhịp.
  - Cần phải có nhiều thanh ghi lệnh để lưu giữ lệnh mà chúng ta phải xem xét cho mỗi giai đoạn thi hành lệnh.
  - Cuối cùng phải có nhiều thanh ghi bộ đếm chương trình PC để có thể tái tục các lệnh trong trường hợp có ngắt quãng.

Pipeline

- Khó khăn trong kỹ thuật ống dẫn
  - Khó khăn do cấu trúc
  - Khó khăn do số liệu
  - Khó khăn do điều khiển

Pipeline

- Khó khăn do cấu trúc
  - Đây là khó khăn do thiếu bộ phận chức năng, ví dụ trong một máy tính dùng kỹ thuật ống dẫn phải có nhiều ALU, nhiều PC, nhiều thanh ghi lệnh IR . Do vậy cần thêm các bộ phận chức năng cần thiết và hữu hiệu.

Pipeline

- Khó khăn do số liệu
  - Toán tử kết quả của lệnh trước có thể chỉ có thể được dùng cho lệnh sau sau giai đoạn MEM của nó, nhưng toán tử được dùng cho lệnh sau vào giai đoạn EX của lệnh trước. Để khắc phục khó khăn này, một bộ phận phần cứng được dùng để đưa kết quả từ ngã ra ALU trực tiếp vô một trong các thanh ghi ngã vào

Pipeline

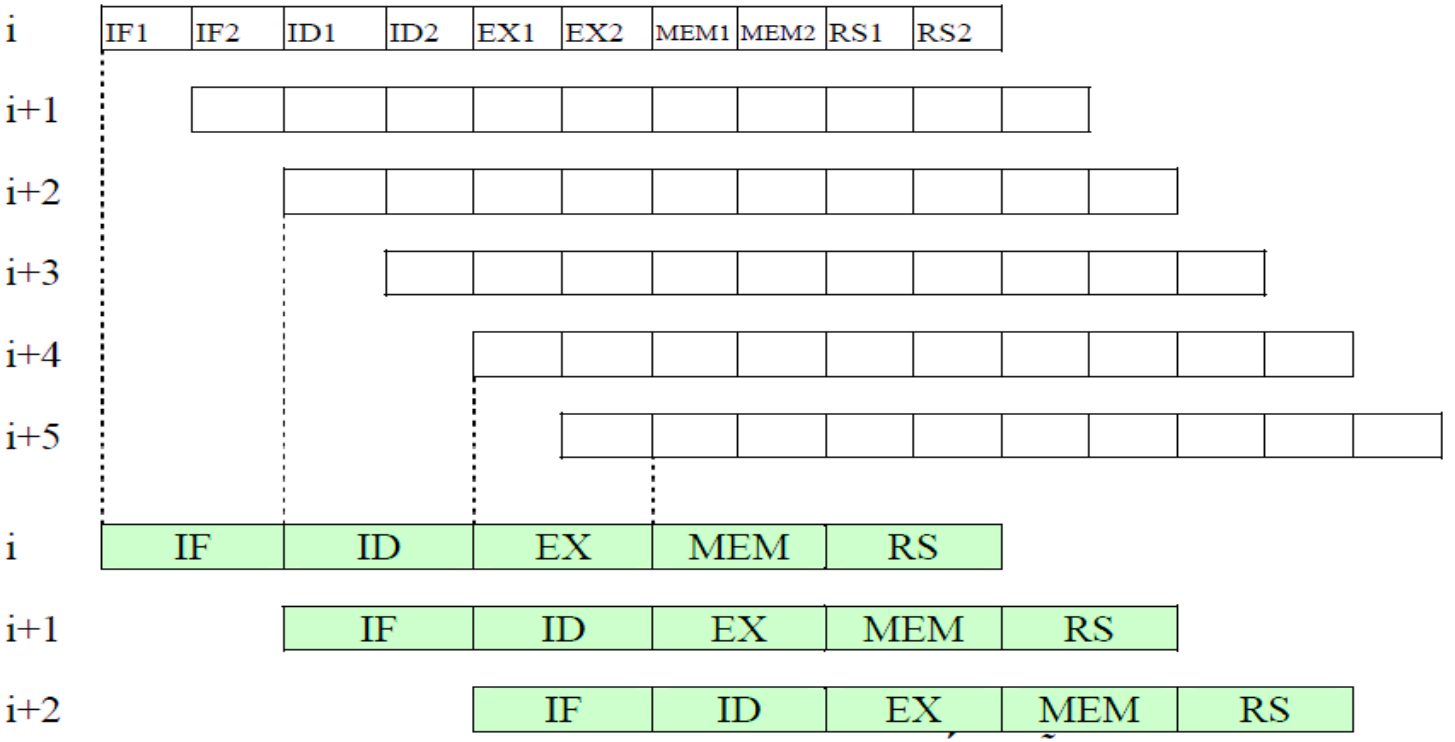
- Khó khăn do điều khiển:
  - Các lệnh làm thay đổi tính thi hành các lệnh một cách tuần tự (nghĩa là PC tăng đều đặn sau mỗi lệnh), gây khó khăn về điều khiển. Các lệnh này là lệnh nhảy đến một địa chỉ tuyệt đối chứa trong một thanh ghi, hay lệnh nhảy đến một địa chỉ xác định một cách tương đối so với địa chỉ hiện tại của bộ đếm chương trình PC. Các lệnh nhảy trên có thể có hoặc không điều kiện.

Hyper Pipeline

- Máy tính có kỹ thuật siêu ống dẫn bậc n:
  - Chia các giai đoạn của kỹ thuật ống dẫn đơn giản, mỗi giai đoạn được thực hiện trong khoản thời gian T_c, thành n giai đoạn con thực hiện trong khoản thời gian Tc/n.
  - Độ hữu hiệu của kỹ thuật này tương đương với việc thi hành n lệnh trong mỗi chu kỳ T_c.

Hyper Pipeline

- Ví dụ về siêu ống dẫn bậc 2
  - Trong khoảng thời gian T_c, máy có siêu ống dẫn làm 2 lệnh thay vì 1 lệnh như trong máy có kỹ thuật ống dẫn đơn giản.



Hyper Pipeline

- Ví dụ về siêu ống dẫn bậc 2
  - Ta thấy trong một chu kỳ Tc, máy dùng kỹ thuật siêu ống dẫn làm 2 lệnh thay vì làm1 lệnh trong máy dùng kỹ thuật ống dẫn bình thường.
  - Trong máy tính siêu ống dẫn, tốc độ thực hiện lệnh tương đương với việc thực hiện một lệnh trong khoảng thời gian Tc/n. Các bất lợi của siêu ống dẫn là thời gian thực hiện một giai đoạn con ngắn Tc/n và việc trì hoãn trong thi hành lệnh nhảy lớn.
  - Nếu lệnh thứ i là một lệnh nhảy tương đối thì lệnh này được giải mã trong giai đoạn ID, địa chỉ nhảy đến được tính vào giai đoạn EX, lệnh phải được nhảy tới là lệnh thứ i+4, vậy có trì trệ 3 lệnh thay vì 1 lệnh trong kỹ thuật ống dẫn bình thường.

Đặc điểm Pipeline

- Pipeline là kỹ thuật song song ở mức lệnh(ILP)
- Một pipeline là trọn vẹn nếu nó luôn nhận một lệnh mới tại mỗi chu kỳ đồng hồ
- Mộtpipeline là không trọnvẹn nếu có nhiều giai đoạn trễ trong quá trình xử lý
- Số lượng giai đoạn của pipeline phụ thuộc vào thiết kế CPU:
  - 5 giaiđoạn: pipeline đơngiản
  - 10 –15 giaiđoạn: Pen III, M, Core 2 Duo
  - 20 giaiđoạn: Pen IV (except Prescott)
  - 31 giaiđoạn: Pen IV Prescott

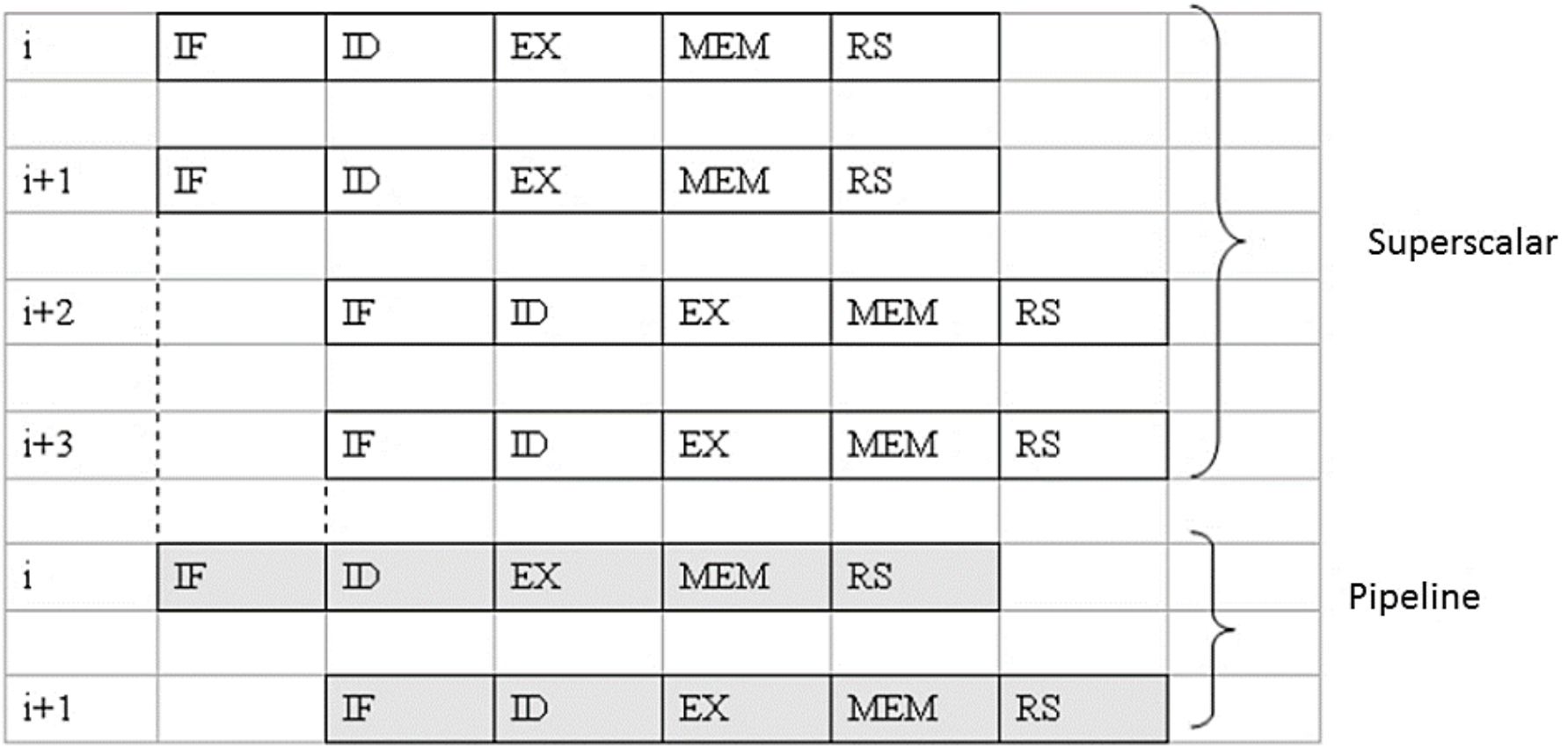
SUPERSCALAR

- Siêu vô hướng
  - Máy tính siêu vô hướng có thể thực hiện đồng thời nhiều lệnh lệnh trong một chu kỳ xung nhịp

SUPERSCALAR

- Trong một máy tính siêu vô hướng phần cứng phải quản lý việc đọc và thi hành đồng thời nhiều lệnh => phải có khả năng quản lý các quan hệ giữa số liệu với nhau.
- Cần phải chọn các lệnh có khả năng được thi hành cùng một lúc.
- Năm 1992 người ta thấy xuất hiện các bộ xử lý có nhiều bộ thực hiện tác vụ độc lập với nhau (nhiều ALU, bộ tính toán số lẻ, nạp dữ liệu, lưu dữ liệu, nhảy), có thể thực hiện song song nhiều lệnh (lệnh tính số nguyên, số lẻ, lệnh bộ nhớ, lệnh nhảy...). Số lệnh có thể được thi hành song song càng nhiều thì phần cứng thực hiện việc này càng phức tạp.
    - Những bộ xử lý đầu tiên đưa ra thị trường dùng kỹ thuật này là các bộ xử lý Intel i860 và IBM RS/6000. Các bộ xử lý này có khả năng thực hiện song song nhiều tác vụ trên số nguyên và trên số lẻ.

SUPERSCALAR

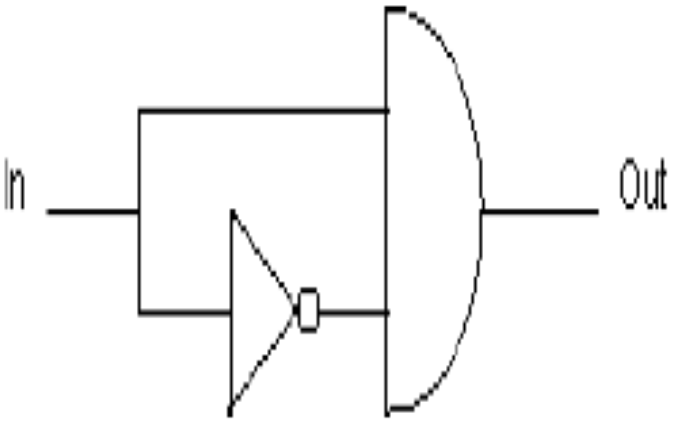


Superscalar

- Trong kiến trúc siêu vô hướng, việc xử lý một lệnh được cắt ra rất nhỏ và nhiều lệnh được xử lý đồng thời miễn là không gây ra tranh chấp dữ liệu.
  - Hai lệnh có tranh chấp dữ liệu là lệnh này có sử dụng kết quả do lệnh kia tạo ra. Trong trường hợp đó bắt buộc phải tôn trọng thứ tự. Sau đó bộ xử lý sẽ liên kết kết quả các xử lý các thành phần.
- Điều phức tạp nhất trong xử lý cả một dãy lệnh còn liên quan tới việc “gọi nhầm” một dãy lệnh từ bộ nhớ lên cache theo thứ tự bình thường do hiện tượng rẽ nhánh

Pipeline hazard

- Đầu ra mong muốn luôn là 0 (false)
- Nhưng trong một số trường hợp, đầu ra là 1 (true) → Hazard

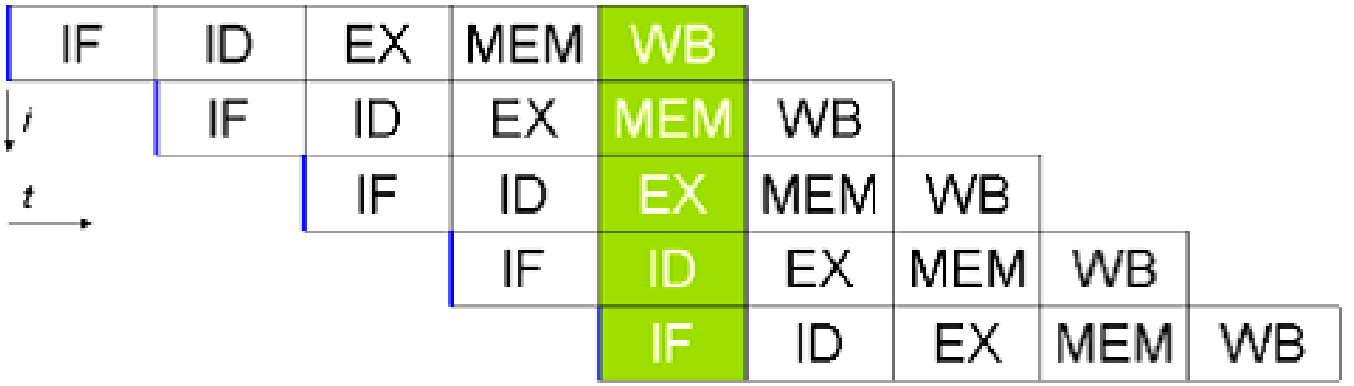


Pipeline hazard

- Xung đột tàinguyên
- Xung đột dữ liệu(hầu hết là RAW hay Read After Write Hazard)
- Các lệnh rẽ nhánh

Pipeline hazard –xung đột tài nguyên

- Tài nguyên không đủ
- Vídụ: nếu bộ nhớ chỉ hỗ trợ một thao tác đọc/ ghi tại một thời điểm, thì các truy cập bộ nhớ đồng thời từ pipeline sẽ gây ra xung đột tài nguyên



Pipeline hazard –xung đột tài nguyên

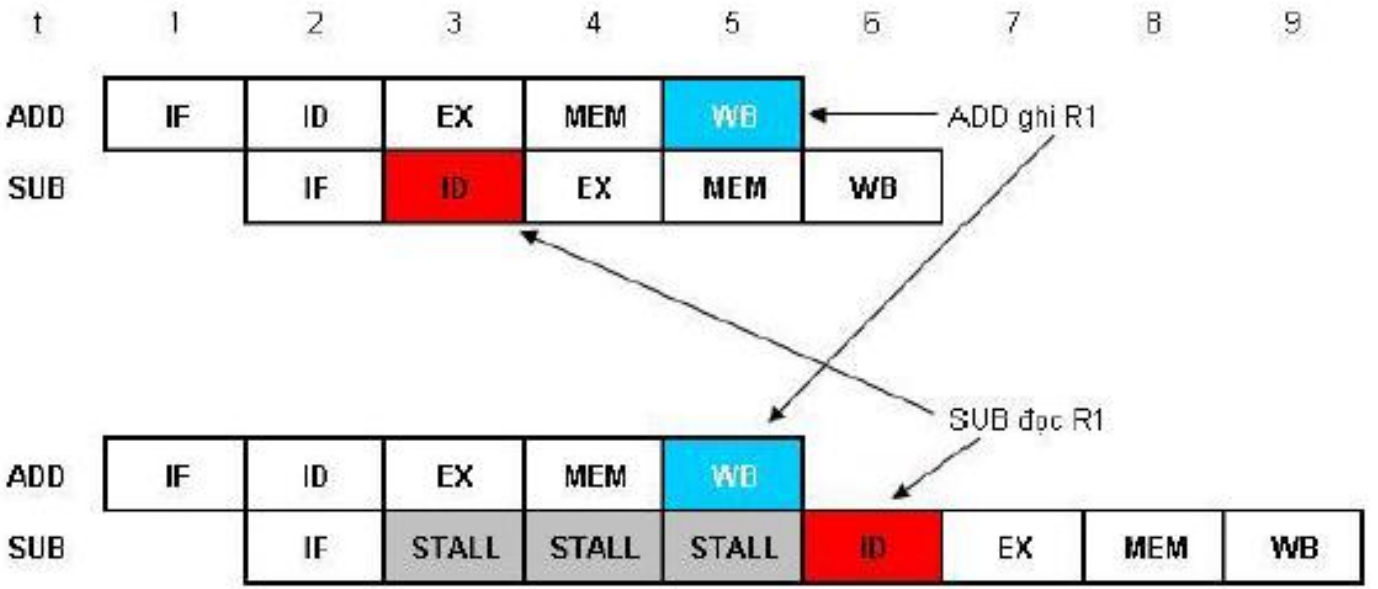
- Giải pháp:
  - Nâng cao khả năng tài nguyên
  - Memory/ cache: hỗ trợ nhiều thao tác đọc/ ghi cùng lúc
  - Chia cache thành cache lệnh và cache dữliệu để cải thiện truy nhập

Pipeline hazard –xung đột dữ liệu

- Xét 2 lệnh sau:
  - ADD R1, R1, R3; R1 ←R1 + R3
  - SUB R4, R1, R2; R4 ←R1 - R2
- SUB sử dụng kết quả lệnh ADD: có phụ thuộc dữ liệu giữa 2 lệnh này
- SUB đọc R1 tại giai đoạn2 (ID); trong khi đó ADD lưu kết quả tại giai đoạn 5 (WB)
  - SUB đọc giá trị cũ của R1 trước khi ADD lưu trữ giá trị mới vào R1

⇒Dữ liệu chưa sẵn sang cho các lệnh phụ thuộc tiếp theo

Pipeline hazard –xung đột dữ liệu



ADD R1, R1, R3; R1 ←R1 + R3
SUB R4, R1, R2; R4 ←R1 - R2

Pipeline hazard –xung đột dữ liệu

- Giải pháp hợp lý:
  - Ngưng pipeline (stall): phải làm trễ hoặc ngưng pipeline bằng cách sử dụng một vài phương pháp tới khi có dữ liệu chính xác
  - Sử dụng complier để nhận biết RAW và:
    - Chèn các lệnh NO-OP (NO-OPeration instruction) vào giữa 2 lệnh có RAW
    - Sắp xếp lại trình tự chương trình và chèn các lệnh độc lập dữ liệu vào vị trí giữa 2 lệnh có RAW
  - Sử dụng phần cứng để xác định RAW (có trong các CPUs hiện đại)

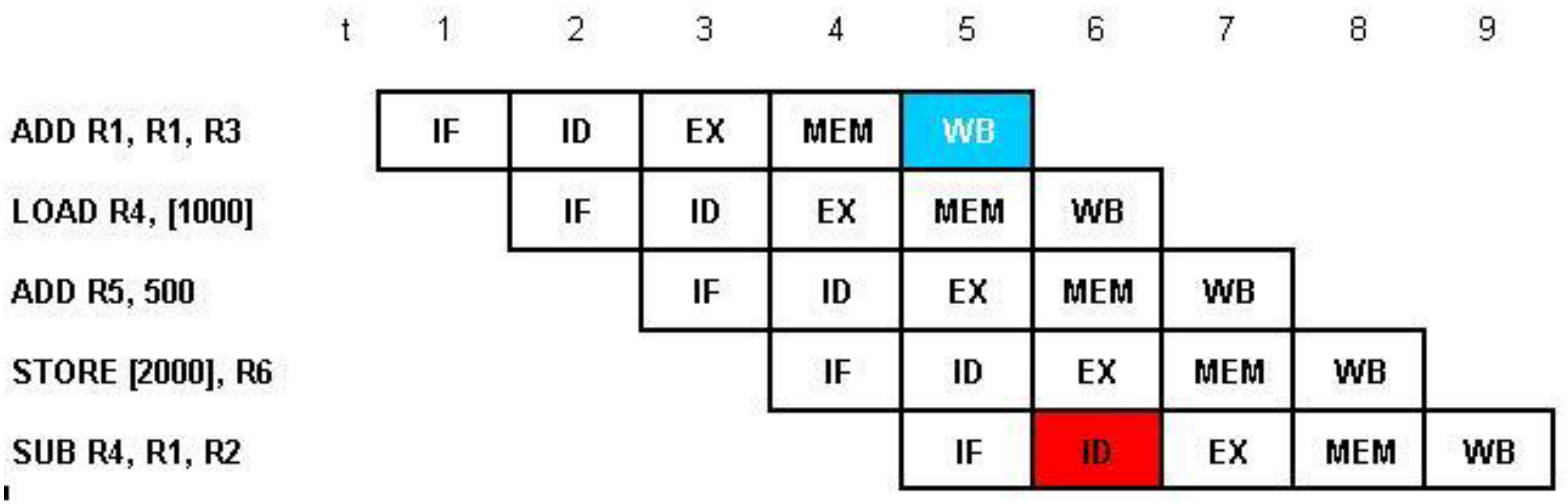
Pipeline hazard –xung đột dữ liệu

- Làm trễ quá trình thực hiện lệnh SUB bằng cách chèn 3 NO-OP

t	1	2	3	4	5	6	7	8	9
ADD	IF	ID	EX	MEM	WB				
NO-OP		NO-OP	NO-OP	NO-OP	NO-OP	NO-OP			
NO-OP			NO-OP	NO-OP	NO-OP	NO-OP	NO-OP		
NO-OP				NO-OP	NO-OP	NO-OP	NO-OP	NO-OP	
SUB					IF	ID	EX	MEM	WB

Pipeline hazard –xung đột dữ liệu

- Chèn 3 lệnh độc lập dữ liệu vào giữa ADD và SUB

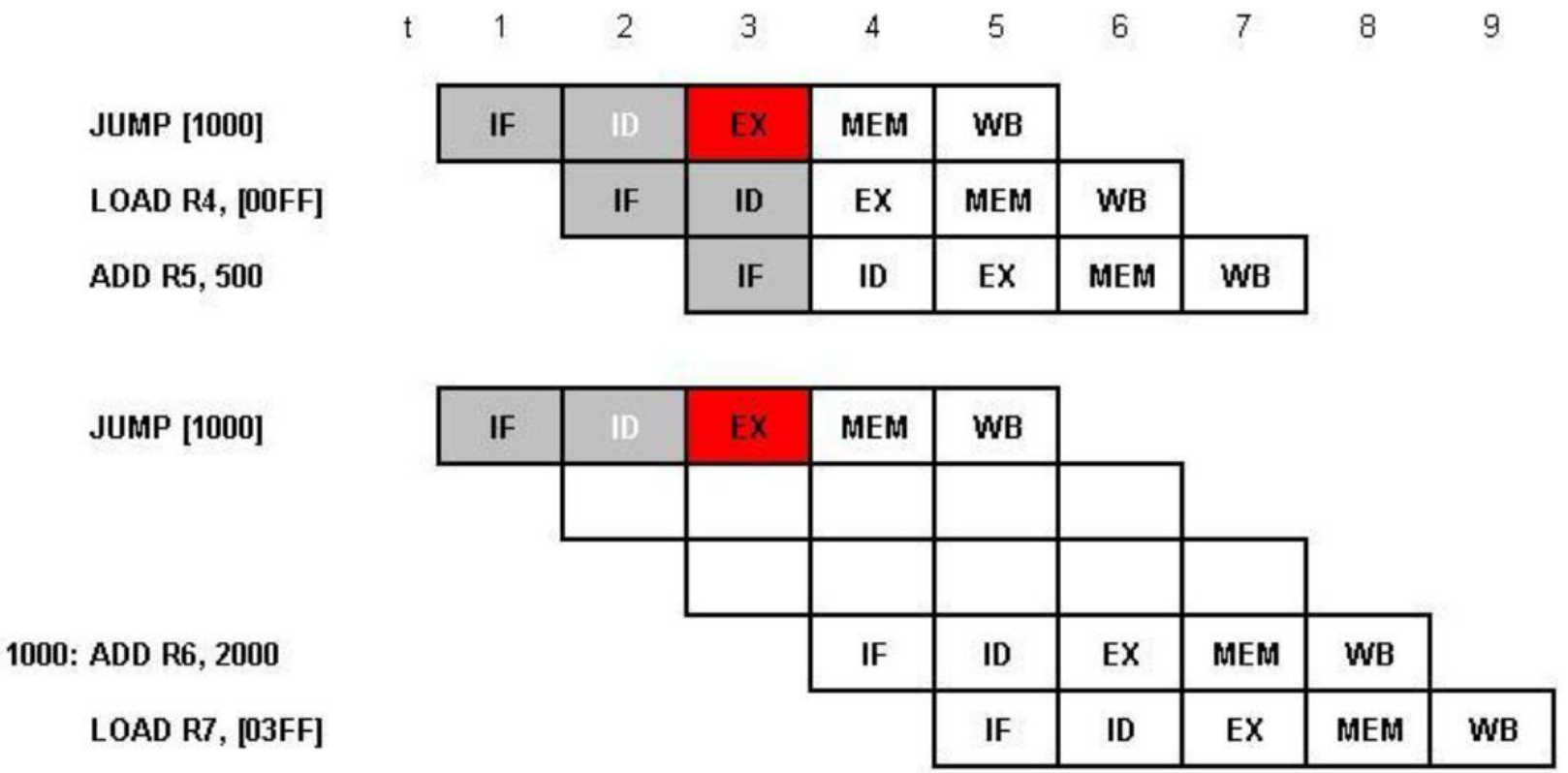


Pipeline hazard – Quản lý các lệnh rẽ nhánh

- Các lệnh rẽ nhánh chiếm khoảng 30%. Các lệnh rẽ nhánh có thể gây ra:
  - Gián đoạn trong quá trình chạy bình thường của chương trình
  - Pipeline rỗng nếu không có giới hạn (thước đo) ngăn chặn hiệu quả
- Với các CPU mà pipeline dài (P4 với 31 giai đoạn) vấn đề rẽ nhánh còn phức tạp hơn vì:
  - Đẩy mọi lệnh đang được xử lý ra ngoài pipeline
  - Tải mới các lệnh vào pipeline

Pipeline hazard – Quản lý các lệnh rẽ nhánh

- Khi 1 lệnh rẽ nhánh được thực hiện, các lệnh tiếp theo bị đẩy ra khỏi pipeline và các lệnh mới được tải



Pipeline hazard – Quản lý các lệnh rẽ nhánh

- Giải pháp
  - Đích rẽ nhánh (target)
  - Rẽ nhánh có điều kiện
    - Làm chậm rẽ nhánh
    - Dự báo rẽ nhánh

Pipeline hazard – Đích rẽ nhánh

- Giải pháp
  - Khi một lệnh rẽ nhánh được thực hiện, lệnh tiếp theo được lấy là lệnh tại đích rẽ nhánh(target) chứ không phải lệnh tại vị trí tiếp theo

    JUMP <Address>ADD R1, R2

    Address: SUB R3, R4

Pipeline hazard – Đích rẽ nhánh

- Giải pháp (tiếp)
  - Các lênh rẽ nhánh được xác định tại giai đoạn ID và có thể được xác định bởi giải mã trước
    - Sử dụng đệm đích rẽ nhánh (BTB: branch target buffer) để lưu trữ:
      - Địa chỉ đích của các lệnh rẽ nhánh được thực hiện
      - Lệnh đích của các lệnh rẽ nhánh được thực hiện
  - Nếu các lệnh rẽ nhánh được sử dụng lại (đặc biệt là lệnh loop):
    - Các địa chỉ đích của chúng lưu trong BTB có thể được dung mà không cần tính lại
    - Các lệnh đích có thể dung trực tiếp không cần load lại từ bộ nhớ

  $\Rightarrow$Điều này có thể vì địa chỉ và lệnh đích thường không thay đổi

Pipeline hazard – Quản lý các lệnh rẽ nhánh

- Lệnh nhảy có điều kiện
  - Khó quản lý các lệnh rẽ nhánh có điều kiện hơn vì:
    - Có 2 lệnh đích để lựa chọn
    - Không thể xác định được lệnh đích tới khi lệnh rẽ nhánh được thực hiện xong
    - Sử dụng BTB không hiệu quả vì phải đợi tới khi có thể xác định được lệnh đích

Pipeline hazard – Làm chậm rẽ nhánh

- Làm chậm rẽ nhánh
  - Dựa trên ý tưởng:
    - Lệnh rẽ nhánh không làm rẽ nhánh ngay lập tức
    - Mà nó sẽ bị làm chậm một vài chukỳ đồng hồ phụ thuộc vào độ dài của pipeline
  - Các đăc điểm của làm chậm rẽ nhánh:
    - Hoạt động tốt trên các vi xử lý RISC trong đó các lệnh có thời gian xử lý bằng nhau
    - Pipeline ngắn (thông thường là 2 giai đoạn)
    - Lệnh sau lệnh nhảy thường được thực hiện không quan tâm tới kết quả lệnh rẽ nhánh

Pipeline hazard – Làm chậm rẽ nhánh

- Sử dụng complier để chèn NO-OP vào vị trí ngay sau lệnh rẽ nhánh, hoặc
- Chuyển một lệnh độc lập từ trước tới ngay sau lệnh rẽ nhánh

Pipeline hazard – Làm chậm rẽ nhánh

- Xét các lệnh:

  ADD R2, R3, R4

  CMP R1,0

  JNE somewhere

- Chèn NO-OP vào vị trí ngay sau lệnh rẽ nhánh

  ADD R2, R3, R4

  CMP R1,0

  JNE somewhere

  NO-OP

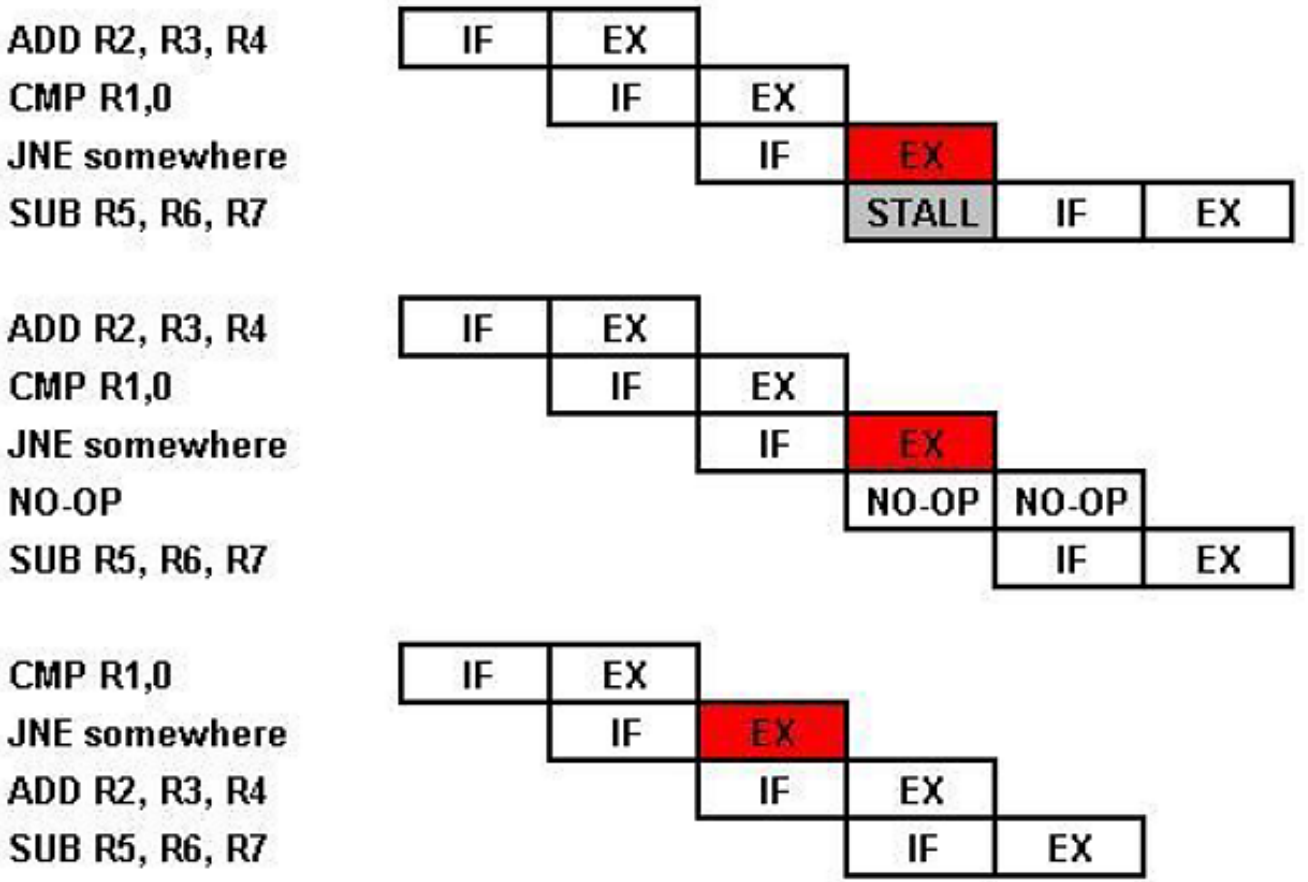
- Chuyển một lệnh độc lập từ trước tới ngay sau lệnh rẽ nhánh

  CMP R1,0

  JNE somewhere

  ADD R2, R3, R4

Pipeline hazard – Làm chậm rẽ nhánh



Pipeline hazard – Làm chậm rẽ nhánh

- Làm chậm rẽ nhánh – các nhận xét
  - Dễ cài đặt nhờ tối ưu trình biên dịch(complier)
  - Không cần phần cứng đặc biệt
  - Chèn NO-OP làm giảm hiệu năng pipeline
  - Thay các lệnh NO-OP bằng các lệnh độc lập có thể làm giảm số lượng NO-OP cần thiết tới 70%

Pipeline hazard – Làm chậm rẽ nhánh

- Làm chậm rẽ nhánh – các nhận xét (tiếp)
  - Làm tăng độ phức tạp mã chương trình (code)
  - Cần lập trình viên và người xây dựng trình biên dịch có mức độ hiểu biết sâu về pipeline vi xửlý. Khó có thể đạt được
  - Giảm khả năng linh hoạt (portable) của mã chương trình vì các chương trình phải được viết và biên dịch lại trên các nềnVXL mới

Pipeline hazard- Branch Prediction

- Một lệnh rẽ nhánh bao giờ cũng liên quan tới một điều kiện được kiểm tra. Nếu kết quả là đúng thì thực hiện một khối lệnh này, sai thì thực hiện khối lệnh kia. Điều không may là, đoạn lệnh được nạp vào trong cache lại không phải đoạn lệnh sẽ phải thực hiện gây ra phải thay thế cache
- Cơ chế dự đoán rẽ nhánh được phát triển gần đây cho phép dự báo rẽ nhánh với xác xuất đúng trên 90% cho phép giảm thiểu việc truy xuất từ bộ nhớ lên cache.

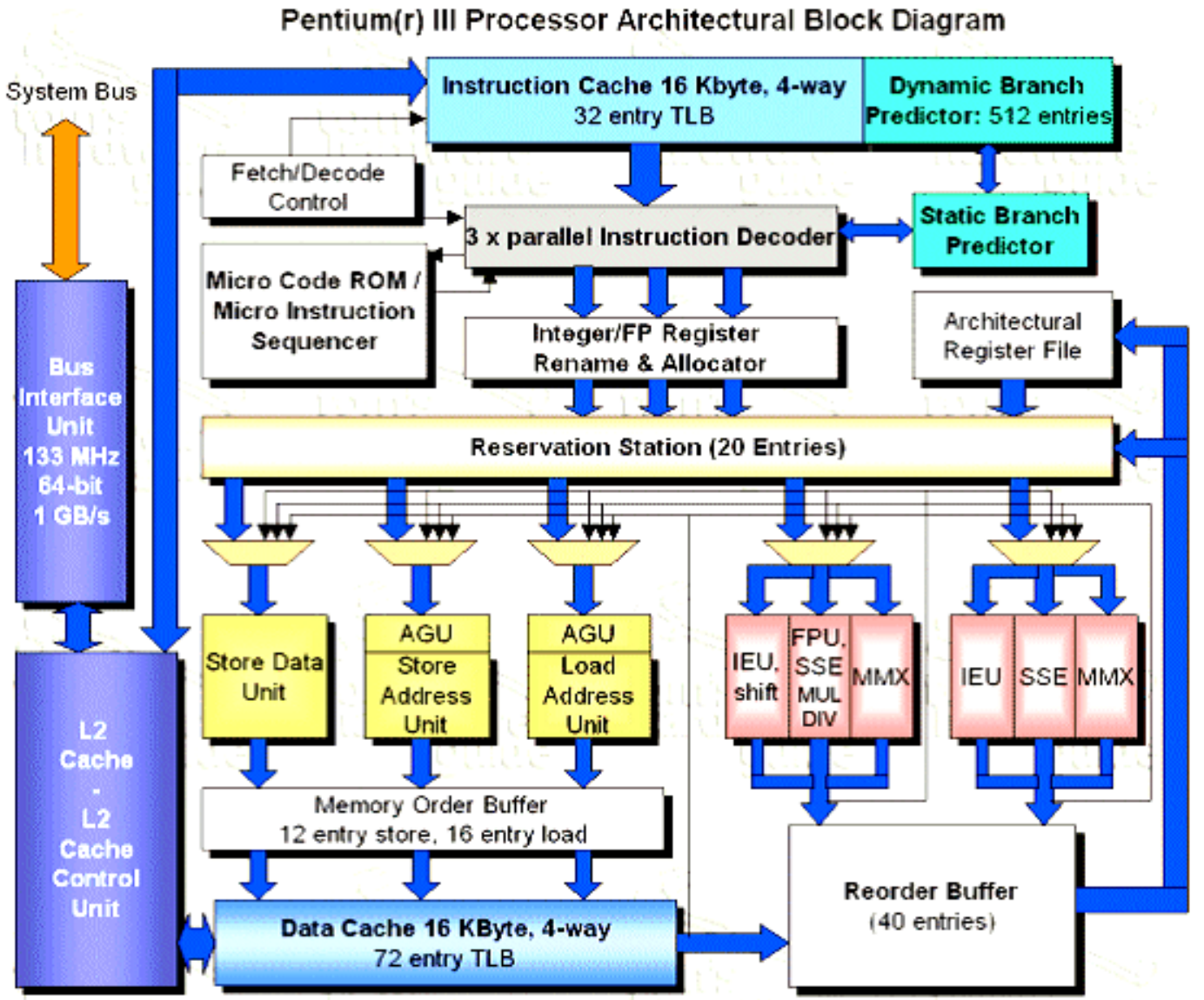
Pipeline hazard- Branch Prediction

- Có thể dự đoán lệnh đích của lệnh rẽ nhánh:
  - Dự đoán đúng: nâng cao hiệu năng
  - Dự đoán sai: đẩy các lệnh tiếp theo đã load và phải load lại các lệnh tại đích rẽ nhánh
  - Trường hợp xấu của dự đoán là 50% đúng và 50% sai

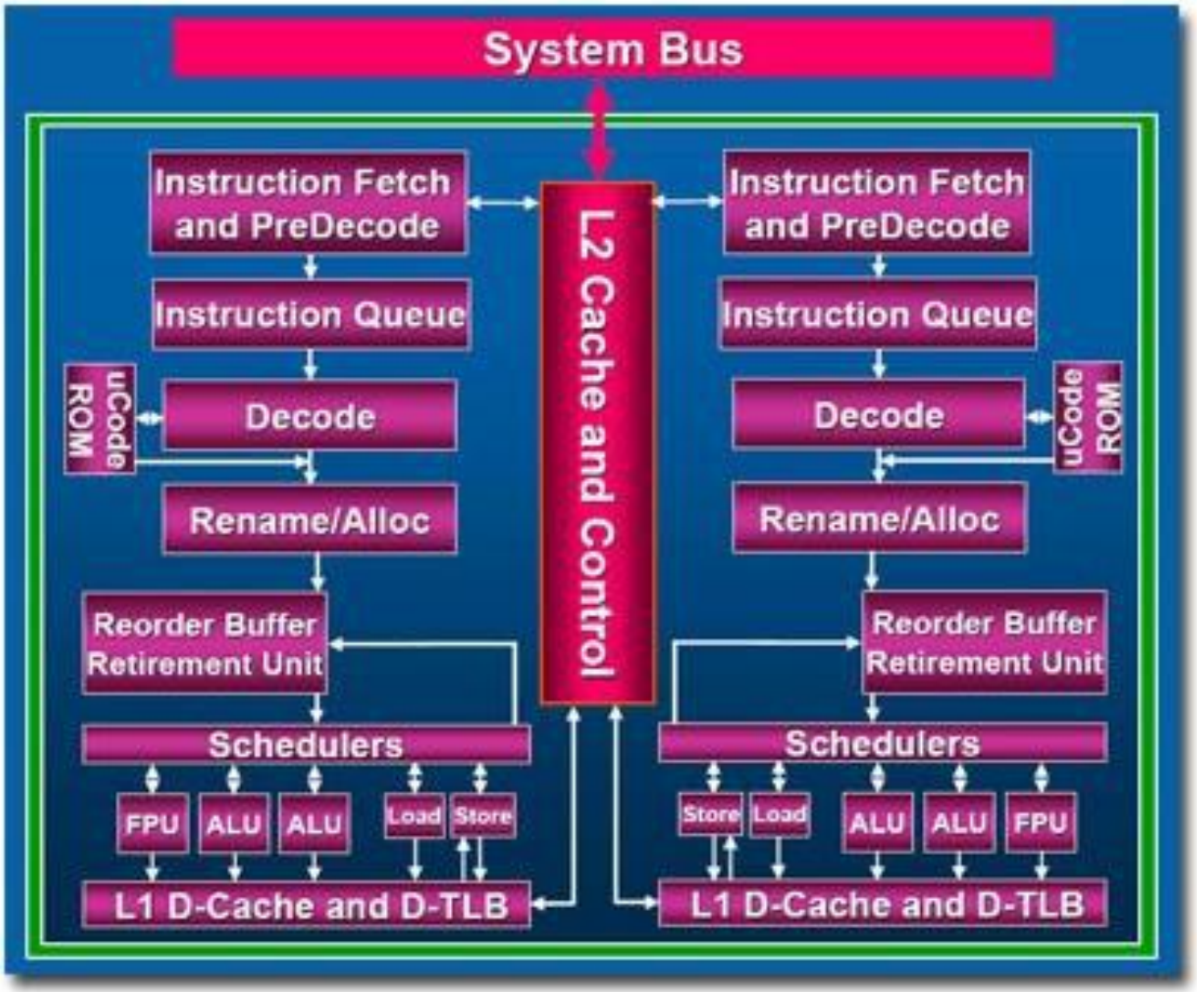
Pipeline hazard - Branch Prediction

- Các căn cứ dự đoán:
  - Đối với các lệnh rẽ nhánh backward:
    - Thường là một phần của lệnh loop
    - Loop thường được thực hiện nhiều lần
  - Đối với các lệnh rẽ nhánh forward:
    - Có thể là kết thúc lệnh loop
    - Có thể là nhảy có điều kiện

Pipeline hazard - Branch Prediction

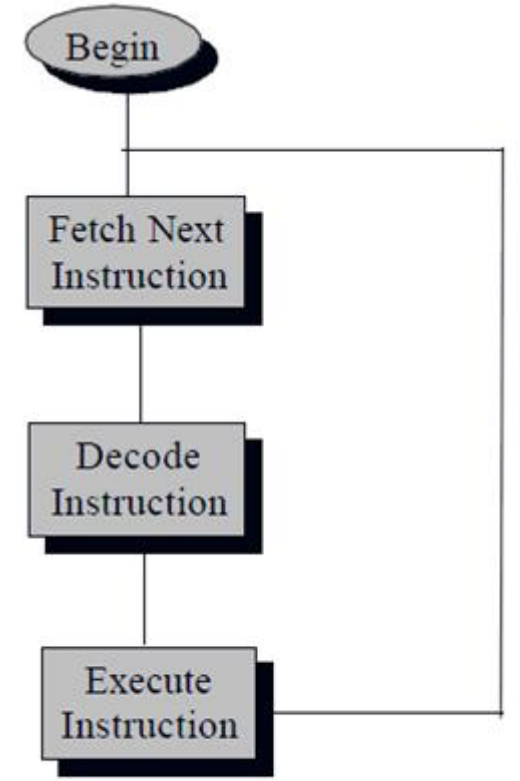


Pipeline hazard - Branch Prediction



Instruction Fetching

- Nhận (nạp lệnh)
  - Máy tính thực hiện một chu trình liên tục :
    - Lấy lệnh kế tiếp từ bộ nhớ
    - Giải mã lệnh
    - Thực hiện các hướng dẫn



Basic Instruction Fetch Execute Cycle

- **Tham khảo chương 3**

Register renaming

- Là một dạng pipeline đề cập đến sự phụ thuộc dữ liệu giữa các lệnh bằng cách đổi tên các toán hạng thanh ghi của chúng.
- Các loại superscalar hiện nay nạp vài lệnh cùng một lúc và phải phân tích xem các lệnh này có phụ thuộc nhau hay không. Nếu các lệnh không phụ thuộc thì có thể chạy cùng lúc, còn nếu không phải chạy tuần tự. Để tăng thêm hiệu suất người ta còn sử dụng cả renaming register
- Là một kỹ thuật nhằm loại bỏ sự phụ thuộc dữ liệu sai phát sinh từ việc tái sử dụng các thanh ghi kiến trúc bằng lệnh dẫn kế tiếp mà không có bất kỳ phụ thuộc dữ liệu thực sự giữa chúng.

Register renaming

- Ví dụ: xét đoạn mã sau
  - Lệnh 4, 5 và 6 không phụ thuộc các lệnh 1, 2, và 3, nhưng bộ xử lý không thể thực hiện lệnh 4 cho đến khi lệnh 3 được thực hiện
  - Hoặc lệnh 3 sẽ viết sai giá trị. Hạn chế này có thể được loại bỏ bằng cách thay đổi tên của các thanh ghi

#	Instruction
1	`R1 = M[1024]`
2	`R1 = R1 + 2`
3	`M[1032] = R1`
4	`R1 = M[2048]`
5	`R1 = R1 + 4`
6	`M[2056] = R1`

Register renaming

- Ví dụ: xét đoạn mã (tiếp)
  - Lệnh 4, 5, và 6 có thể được thực hiện song song với các lệnh 1, 2, và 3, để chương trình có thể được thực hiện nhanh hơn.

#	Instruction	#	Instruction
1	`R1 = M[1024]`	4	`R2 = M[2048]`
2	`R1 = R1 + 2`	5	`R2 = R2 + 4`
3	`M[1032] = R1`	6	`M[2056] = R2`

- Khi có thể, trình biên dịch sẽ phát hiện các lệnh phân biệt và cố gắng gán chúng cho một thanh ghi khác nhau.
- Tuy nhiên, có một số hữu hạn các tên thanh ghi có thể được sử dụng trong ngôn ngữ máy. Nhiều CPU có hiệu suất cao có nhiều thanh ghi vật lý hơn là có thể đặt tên trực tiếp trong tập lệnh, do đó, có thể đổi tên thanh ghi trong phần cứng để đạt được song song bổ sung

Multiprocessor

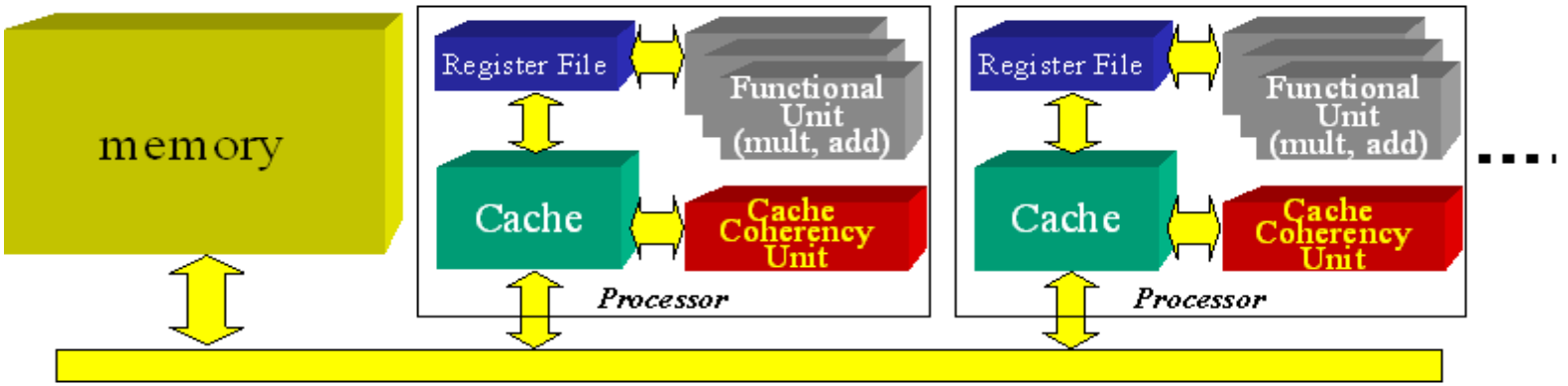
- Đa vi xử lý (hoặc nhiều vi xử lý)
- Là một hệ thống máy tính với hai hoặc nhiều đơn vị xử lý trung tâm (CPU), mỗi đơn vị này chia sẻ bộ nhớ chính chung cũng như các thiết bị ngoại vi. Điều này giúp xử lý đồng thời các chương trình.
- Mục tiêu chính của việc sử dụng Multiprocessor là để tăng tốc độ thực thi của hệ thống, khả năng chịu lỗi và phù hợp với ứng dụng.

Multiprocessor

- Đa vi xử lý (hoặc nhiều vi xử lý)
  - Multiprocessor được coi là một phương tiện để cải thiện tốc độ máy tính, hiệu quả và chi phí-hiệu quả, cũng như để cung cấp tăng cường tính sẵn sàng và độ tin cậy.
  - Trong một hệ thống đa xử lý, tất cả các CPU có thể được bình đẳng, hoặc một số có thể được dành cho các mục đích đặc biệt

Multiprocessor

Multiprocessor Architecture



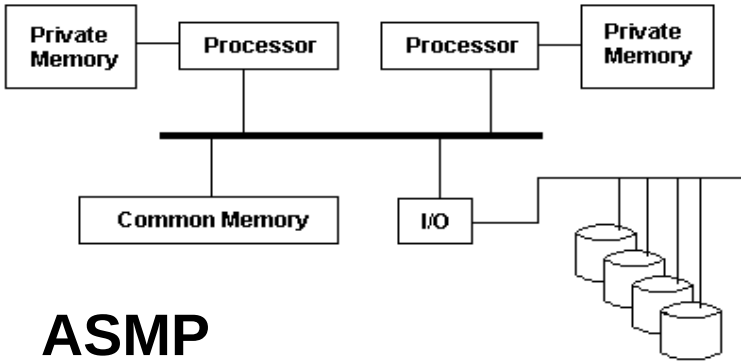
Cache coherency unit will intervene if two or more processors attempt to update same cache line

- All memory (and I/O) is shared by all processors
- Read/write conflicts between processors on the same memory location are resolved by *cache coherency unit*
- Programming model is an extension of single processor programming model

Multiprocessor

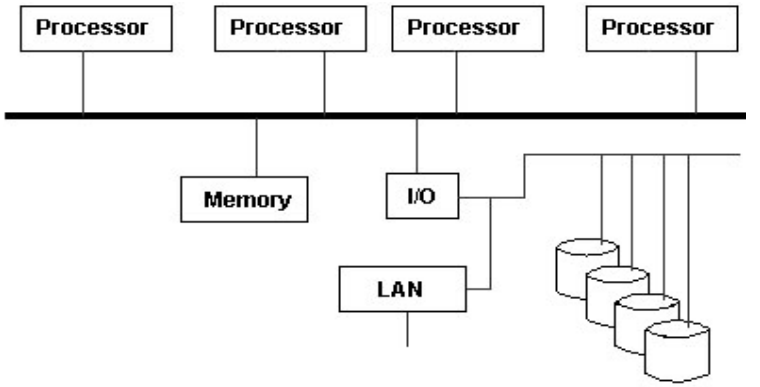
- Các hệ thống xử lý tất cả các CPU một cách ngang bằng đều được gọi là hệ thống đa xử lý đối xứng (SMP symmetric multiprocessing).
- Trong các hệ thống mà tất cả các CPU không ngang bằng nhau, tài nguyên hệ thống có thể được chia thành một số cách khác nhau: không đối xứng (Asymmetric multiprocessing, ASMP) , truy cập bộ nhớ không đông bộ (Non-uniform memory access, NUMA)

Multiprocessor

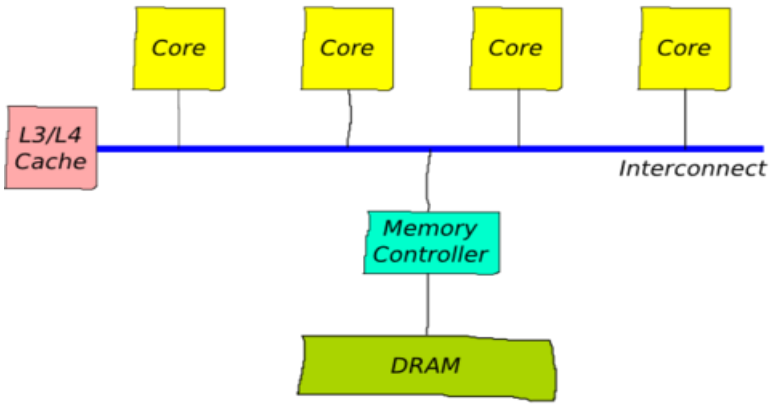


ASMP

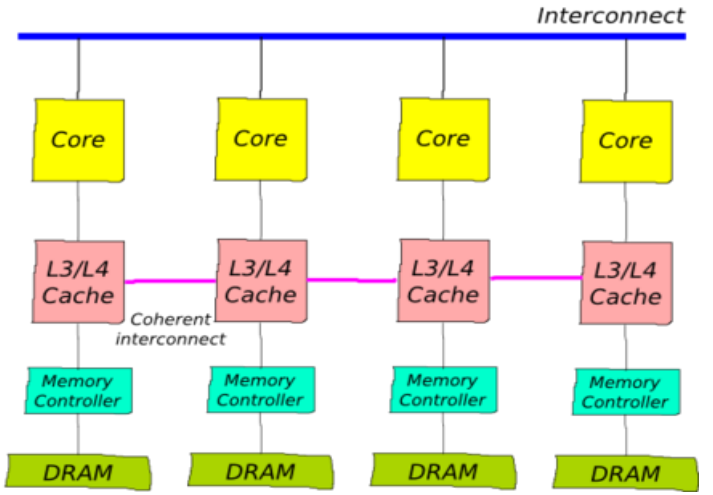
SMP



SMP

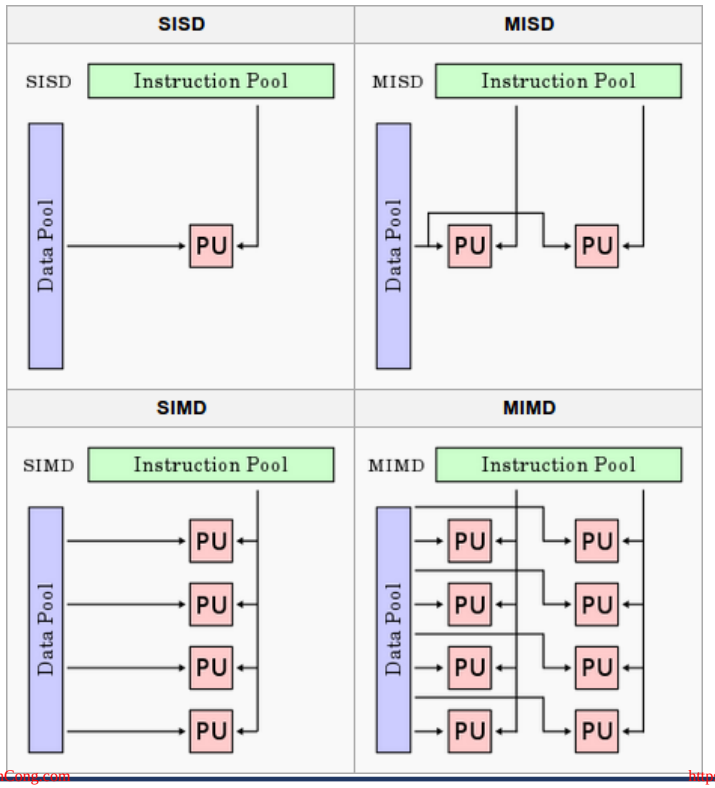


NUMA



Phân loại kiến trúc máy tính

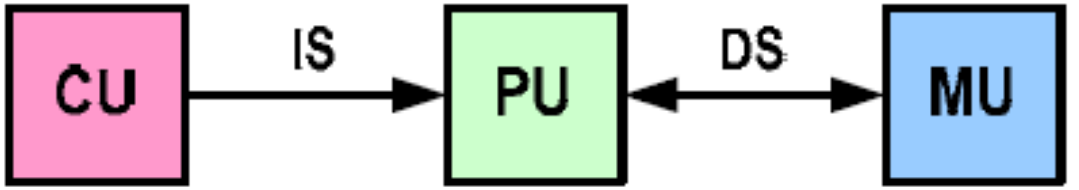
- SISD - Single Instruction Stream, Single Data Stream
- SIMD - Single Instruction Stream, Multiple Data Stream
- MISD - Multiple Instruction Stream, Single Data Stream
- MIMD - Multiple Instruction Stream, Multiple Data Stream

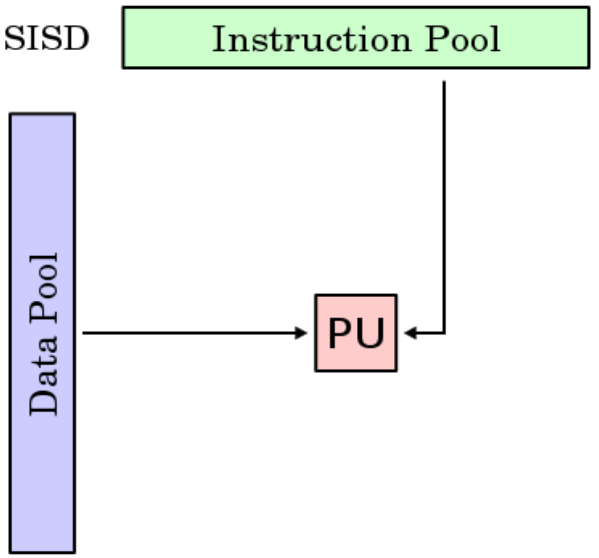


SISD - Single Instruction Stream, Single Data Stream

- CU: Control Unit
- PU: Processing Unit
- MU: Memory Unit
- Một bộ xử lý
- Đơn dòng lệnh
- Dữ liệu được lưu trữ trong một bộ nhớ

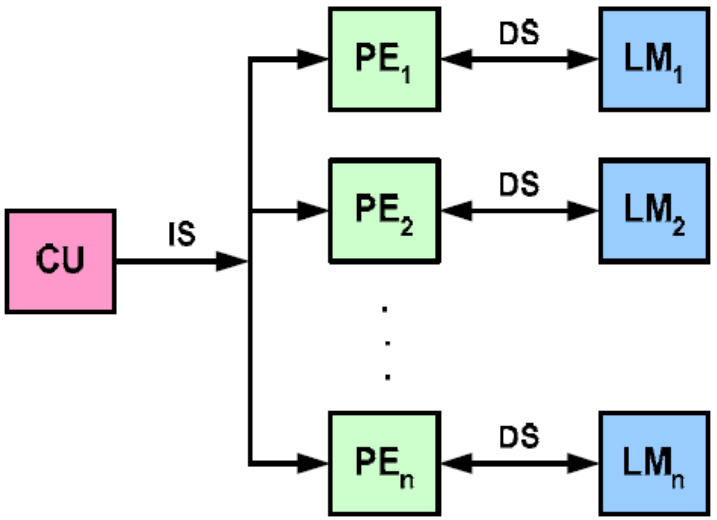
⇒Chính là Kiến trúc von Neumann

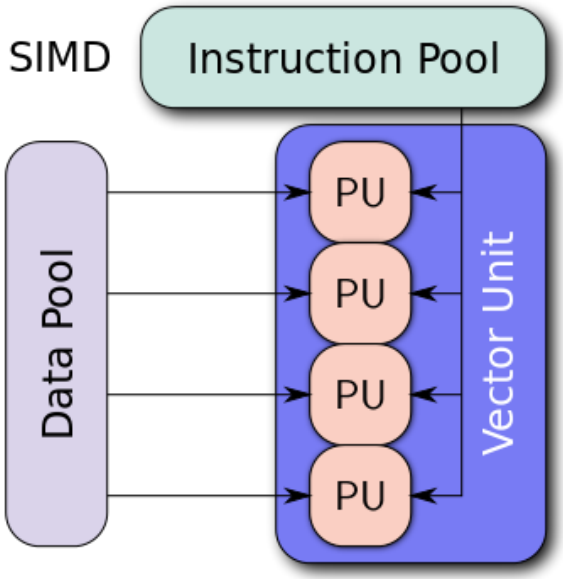




SIMD-Single Instruction Stream, Multiple Data Stream

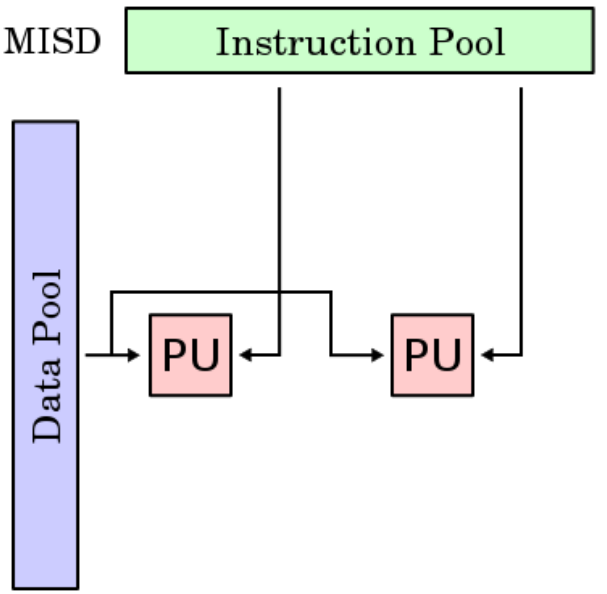
- Đơn dòng lệnh điều khiển đồng thời các phần tử xử lý PE (processing elements)
- Mỗi phần tử xử lý có một bộ nhớ dữ liệu riêng LM (local memory)
- Mỗi lệnh được thực hiện trên một tập các dữ liệu khác nhau





MISD - Multiple Instruction Stream, Single Data Stream

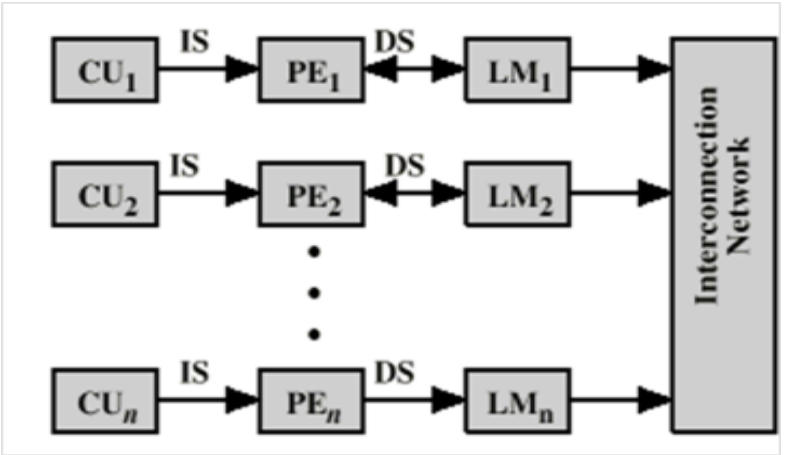
- Một luồng dữ liệu cùng được truyền đến một tập các bộ xử lý
- Mỗi bộ xử lý thực hiện một dãy lệnh khác nhau.
- Không tồn tại máy tính thực tế
- Có thể có trong tương lai
  - Space Shuttle flight control computers (các máy tính điều khiển chuyến bay của tàu con thoi)?



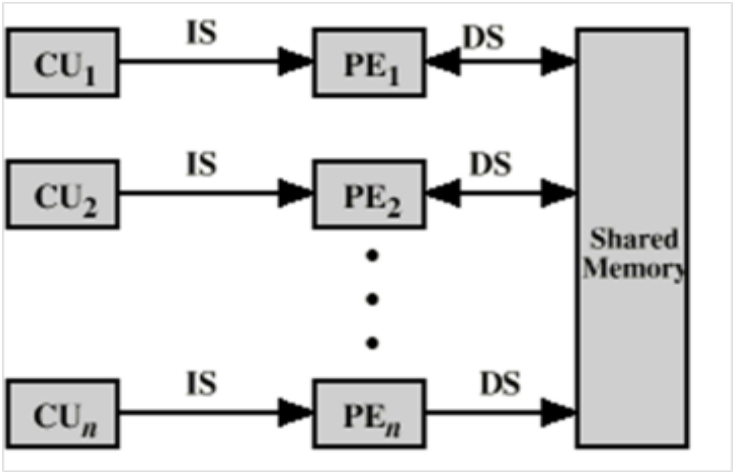
MIMD - Multiple Instruction Stream, Multiple Data Stream

- Tập các bộ xử lý
- Các bộ xử lý đồng thời thực hiện các dãy lệnh khác nhau trên các dữ liệu khác nhau
- Các mô hình MIMD
  - Multiprocessors (Shared Memory)
  - Multicomputers (Distributed Memory)

MIMD - Multiple Instruction Stream, Multiple Data Stream



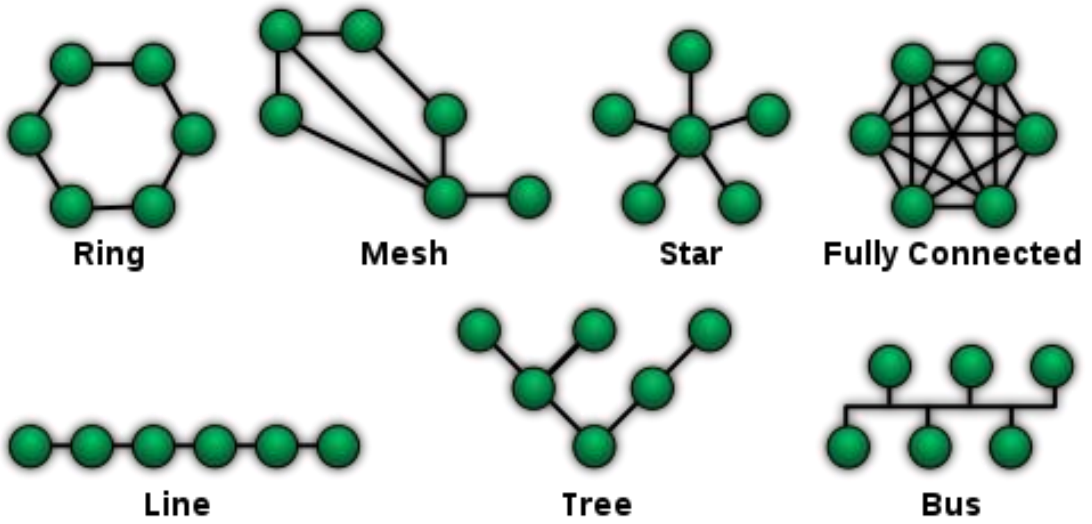
Distributed Memory



Shared Memory

Mạng kết nối

- Mạng máy tính hay hệ thống mạng (computer network hay network system)
  - Là sự kết hợp các máy tính lại với nhau thông qua các thiết bị nối kết mạng và phương tiện truyền thông (giao thức mạng, môi trường truyền dẫn) theo một cấu trúc nào đó và các máy tính này trao đổi thông tin qua lại với nhau.



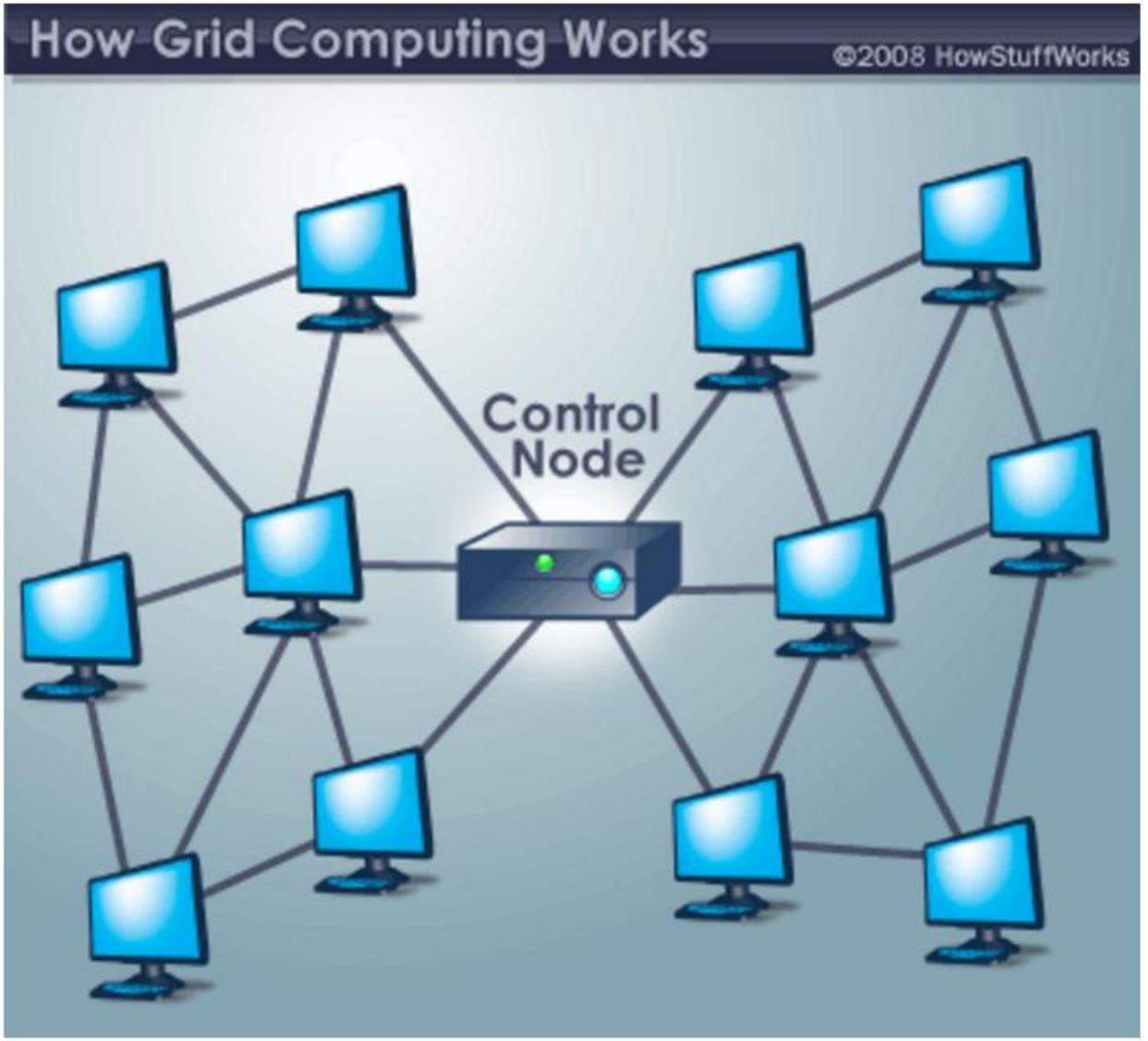
Kiến trúc siêu máy tính

- Những chiếc supercomputer thuở đầu của Seymour Cray hoạt động dựa trên kĩ thuật tính toán song song và thiết kế nhỏ gọn để có thể đạt được hiệu năng tính toán cao.
- Điện toán song song (parellel computing), đây là khái niệm dùng để chỉ việc sử dụng một số lượng lớn CPU để thực hiện một bộ các phép tính nào đó. Tất cả các phép tính sẽ được thực thi song song nhau.
- Có hai cách mà người ta thường áp dụng:
  - grid computing
  - computer cluster

Kiến trúc siêu máy tính- Grid computing

- Sử dụng một mạng lưới nhiều máy tính phân tán ở nhiều nơi để xử lí số liệu (khoảng cách địa lý của chúng tương đối xa nhau)
  - Thường thì khi một chiếc máy tính trong mạng lưới được chạy lên, nó sẽ ngay lập tức trở thành một phần của hệ thống tính toán song song, và càng nhiều máy tính tham gia thì tốc độ xử lí sẽ nhanh hơn.
  - Có một máy chính (Control Node) nằm ở giữa làm nhiệm vụ điều khiển và phân bổ tác vụ cho các máy con.

Kiến trúc siêu máy tính- Grid computing



Kiến trúc siêu máy tính- computer cluster

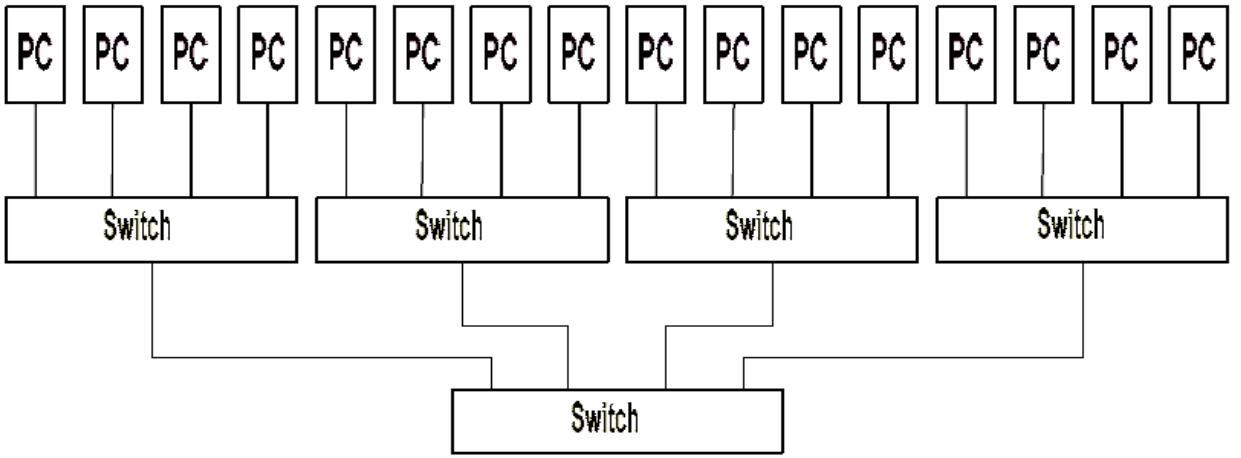
- Sử dụng một lượng lớn CPU đặt gần nhau (Kiểu điện toán tập trung)
  - Những CPU này thường nằm trong nhiều máy tính giống nhau, lân cận nhau (gọi là các node, node card hay computer node) và chúng được kết nối nhằm tạo ra một hệ thống lớn hơn, hoàn chỉnh hơn.
  - Người ta xem cả hệ thống như một siêu máy tính duy nhất. Với biện pháp này, các nhà thiết kế cần đảm bảo rằng tốc độ cũng như tính linh hoạt nội liên kết giữa các máy tính phải đủ đáp ứng yêu cầu công việc.
  - Theo số liệu từ TOP500, số siêu máy tính cluster hiện chiếm đến 82,2% thị phần siêu máy tính toàn cầu. Siêu máy tính IBM Blue Gene/Q sử dụng dạng cluster.

Kiến trúc siêu máy tính- computer cluster

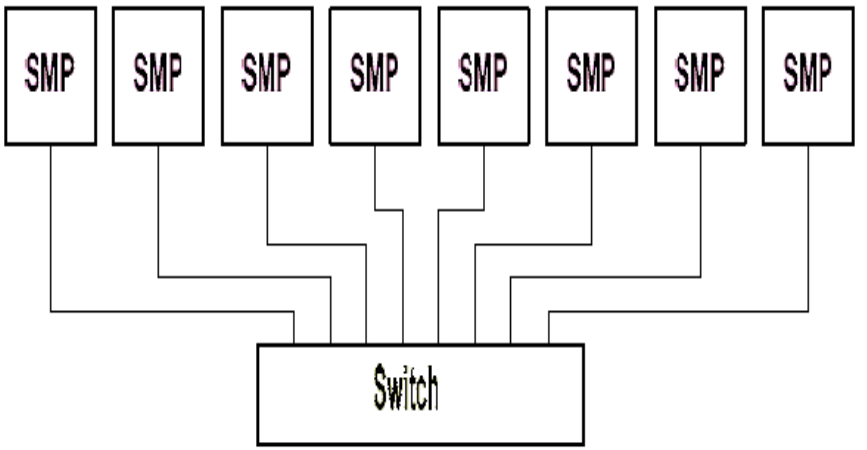
- Nhiều máy tính được kết nối với nhau bằng mạng liên kết tốc độ cao (Fast ethernet hay Gbps)
- Mỗi máy tính (node) có thể làm việc độc lập
- Mỗi máy tính được gọi là một node
- Các máy tính có thể được quản lý làm việc song song theo nhóm (cluster)
- Toàn bộ hệ thống có thể coi như là một máy tính song song

Kiến trúc siêu máy tính- computer cluster

Cluster of PCs

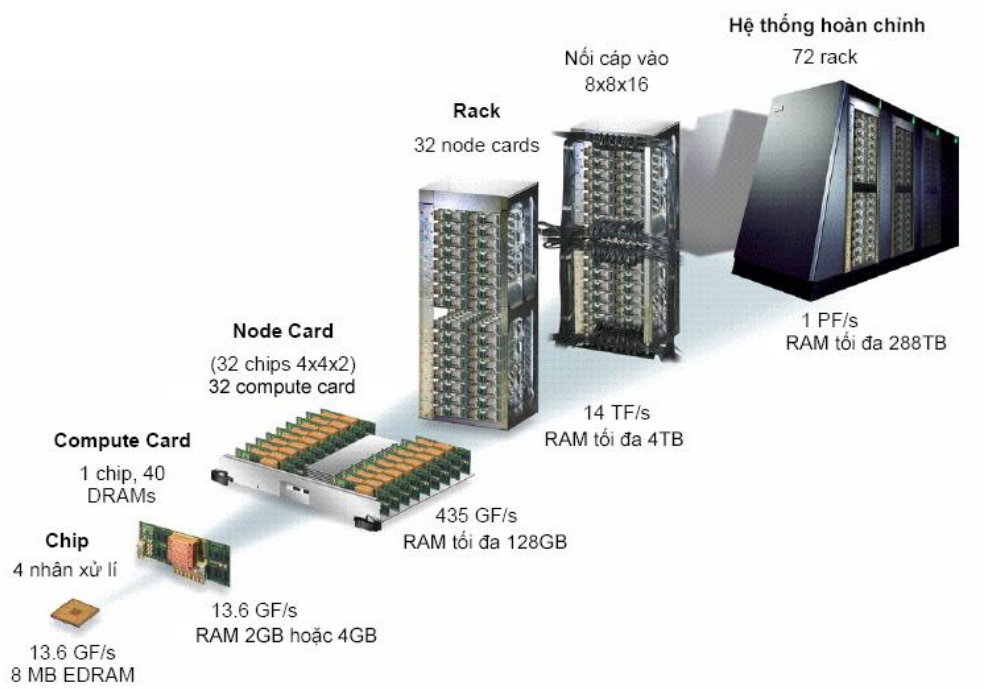


Cluster of SMP



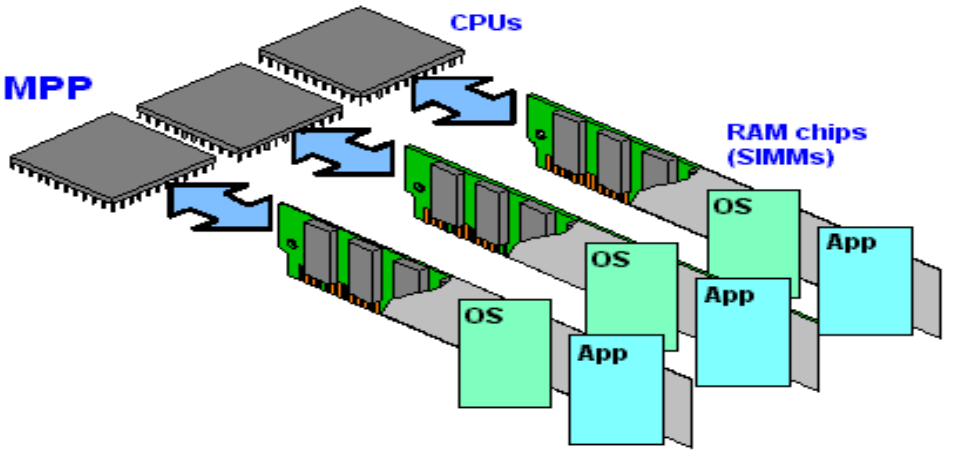
Kiến trúc siêu máy tính- computer cluster

- Từ các vi xử lí, các nhà sản xuất/cung ứng siêu máy tính sẽ tích hợp nó lên một chiếc "computer card" với RAM và có thể có thêm nhiều linh kiện khác để tạo ra một chiếc máy tính gần hoàn chỉnh. Nhiều computer card gắn vào một khay, sau đó các khay lại tiếp được bỏ trong những tủ chứa (rack). Nhiều rack sẽ cấu thành một hệ thống siêu máy tính hoàn chỉnh



Kiến trúc siêu máy tính

- Siêu máy tính dạng Massively Parallel Processors (MPP), tức là một máy tính bự
  - Có hàng nghìn CPU và thanh RAM trong đó. Chúng được nối với nhau theo một chuẩn mạng đặc biệt tốc độ siêu cao chứ không xài các thứ phổ thông như cluster.
  - Mỗi CPU sẽ có bộ nhớ riêng của nó và một bản sao hệ điều hành/ứng dụng riêng. MPP hiện chiếm 17,8% thị phần siêu máy tính, theo TOP500. Siêu máy tính IBM Blue Gene/L (đứng thứ 5 thế giới vào năm 2009) được thiết kế dạng MPP.



Phần cứng siêu máy tính

- Số CPU vài chục nghìn là chuyện hết sức bình thường đối với một siêu máy tính
- Cần đến ổ lưu trữ, và các ổ HDD, SSD này không nằm trong máy (device attached storage - DAS) như trên PC.
    - Chúng thường được bố trí trong một tủ riêng (storage area network - SAN), có kết nối mạng riêng và dung lượng cũng rất "khủng".

Phần cứng siêu máy tính

- Ở các siêu máy tính dạng cluster,
  - Thường ghép nhiều "nút" (node) nhỏ lại với nhau để tạo ra một hệ thống lớn như hình bạn đã thấy ở ngay bên trên.
  - Mỗi một nút như thế có thể xem là một chiếc máy tính gần như hoàn chỉnh với một hoặc nhiều CPU, GPU, nhiều thanh RAM, quạt tản nhiệt và một số thành phần khác nữa.
  - Các nút sẽ được kết nối với nhau theo nhiều cách, có thể là dùng cáp đồng bình thường, cũng có thể là chuyển sang dùng cáp quang để đảm bảo băng thông tốt hơn.
  - Sức mạnh của siêu máy tính sẽ là sức mạnh tổng hợp từ tất cả các node lại với nhau.

Phần cứng siêu máy tính

- Ngoài CPU đơn thuần, người ta còn sử dụng thêm các GPGPU (general purpose graphic processor unit) để tăng cường sức mạnh cho siêu máy tinh
- GPU được sử dụng để dựng hình ảnh, xử lí những thứ liên quan đến đồ họa, nhưng ngoài ra chúng còn có thể xử lí số liệu và làm một số công việc tương tự như CPU.
- Hiện nay giá thành của các GPGPU đã giảm, hiệu suất lại tăng cao nên ngày càng nhiều siêu máy tính "nhờ vả" vào linh kiện này để tăng sức mạnh nhưng vẫn đảm bảo chi phí không bị đội lên quá nhiều.

Phần cứng siêu máy tính

- Siêu máy tính mạnh nhất thế giới (tính đến 5/2013) mang tên Titan đặt ở Mỹ cũng dùng 18.688 GPU NVIDIA Tesla K20 bên cạnh 18.688 CPU AMD Opteron 16 nhân, như vậy tổng số nhân CPU của máy là 299.008, một con số khổng lồ so với máy tính chỉ 2 hay 4 nhân.

Một số tủ thuộc Titan

Phần cứng siêu máy tính

- Một node của siêu máy tính Titan, trong này có 4 CPU AMD Opteron 16 nhân và 4 GPU NVIDIA Tesla K20 (kiến trúc xây dựng là Kepler). RAM mỗi node là 32GB, cộng với 6GB bộ nhớ GDDR5 của card đồ họa nữa. Tổng cộng có 710 Terabyte bộ nhớ trong Titan.

Đo đạc hiệu năng siêu máy tính

- Máy tính bình thường thì đo bằng MIPS (instructions per second - số chỉ dẫn được thực hiện trong mỗi giây). FLOPS có thể được thêm một số tiếp đầu ngữ trong hệ đo lường SI như tera- (TFLOPS, tức 10^{12} FLOPS, đọc là teraflops), peta (10^{15} FLOPS).
- Khả năng tính toán của siêu máy tính được đo bằng FLOPS (FLoating Point Operations Per Second - phép tính dấu chấm động thực hiện trong mỗi giây),

Năm	Tên siêu máy tính	Tốc độ Rmax	Địa điểm
2008	IBM Roadrunner	1.026 PFLOPS	Los Alamos, Mỹ
		1.105 PFLOPS	
2009	Cray Jaguar	1.759 PFLOPS	Oak Ridge, Mỹ
2010	Tianhe-IA	2.566 PFLOPS	Tianjin, Trung Quốc
2011	Fujitsu K computer	10.51 PFLOPS	Kobe, Nhật
2012	IBM Sequoia	16.32 PFLOPS	Livermore, Mỹ
2012	Cray Titan	17.59 PFLOPS	Oak Ridge, Mỹ

Đo đạc hiệu năng siêu máy tính

- Các siêu máy tính hàng đầu thế giới đều đã bước sang ngưỡng Petaflops
  - IBM Roadrunner năm 2008 là 1,105 Petaflops, Fujitsu K năm 2011 đạt mức 10,51 Petaflops, còn chiếc Cray Titan mạnh nhất hiện nay là 17.59 Petaflops. Người ta dự đoán là chỉ sau khoảng 10 năm nữa, siêu máy tính sẽ sớm bước sang hàng Exaflops (10^{18} FLOPS) vì công nghệ của CPU và GPGPU đang tăng trưởng vượt bậc, giá thành lại rẻ đi và hiệu năng tiêu thụ điện ngày càng được nâng cao
- Những con số FLOPS được đo bằng một phần mềm tên là Linpack.

Đo đạc hiệu năng siêu máy tính

- Tuy nhiên cũng cần phải nói thêm rằng không một con số đứng riêng lẻ nào có thể phản ánh toàn bộ hiệu năng của máy tính nói chung và siêu máy tính nói riêng.
- Có hai con số được thể hiện khi nói tới siêu máy tính:
  - Hiệu năng tính toán dấu chấm động lý thuyết của vi xử lí (kí hiệu Rpeak)
    - Rpeak gần như không thể nào đạt được trong đời thực
  - Hiệu năng xử lí đầu vào (Rmax)
    - Rmax là hoàn toàn có thể đạt đến khi siêu máy tính chạy. Tất cả những con số FLOPS ở trên đều là Rmax.

Một số ứng dụng của siêu máy tính

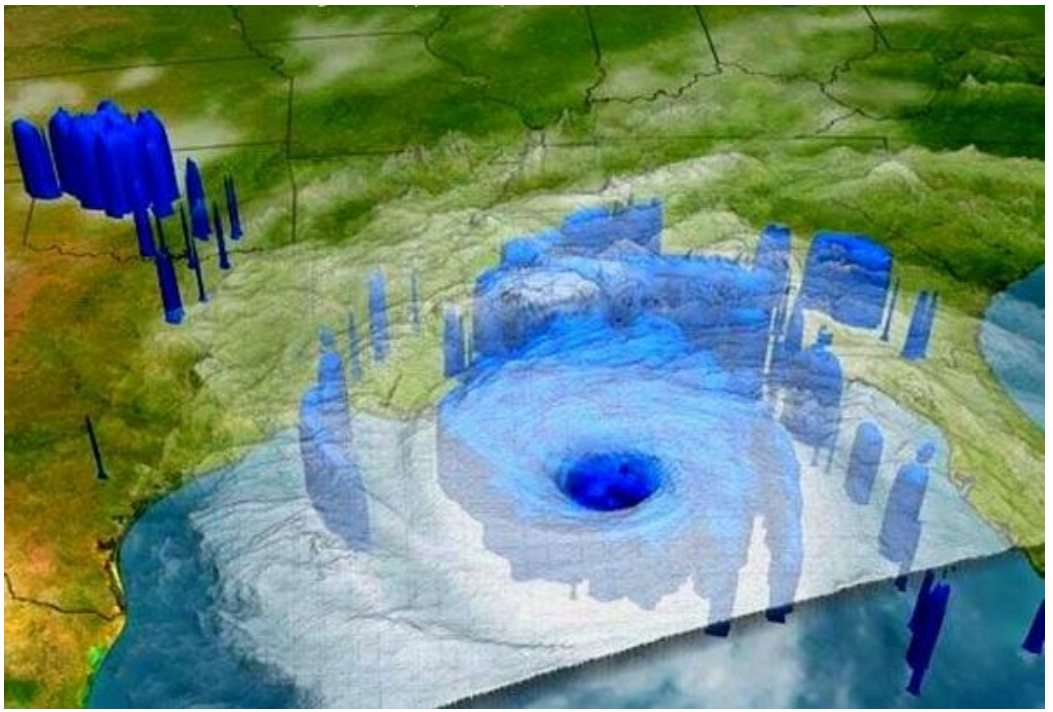
- Dự báo thời tiết, nghiên cứu khí động học, nghiên cứu sự biến đổi khí hậu, mô phỏng động đất
- Phân tích xác suất, dựng mô hình phóng xạ
- Mô phỏng vụ nổ hạt nhân trong không gian 3D
- Lượng tử học, phân tử học, sinh học tế bào, nghiên cứu sự gấp khúc của protein
- Mô phỏng não người
- Nghiên cứu và dựng mô hình của các hiện tượng vật lý

Một số ứng dụng của siêu máy tính

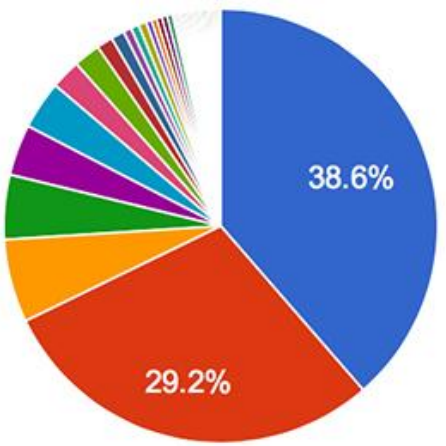
- Nghiên cứu và mô phỏng trí tuệ nhân tạo
- Tái tạo vụ nổ Bigbang (do siêu máy tính ở trung tâm Texas Advanced Computing Center thực hiện), nghiên cứu về vật chất tối
- Nghiên cứu thiên văn học
- Dựng mô hình lây lan của dịch bệnh
- Chơi cờ vua! (siêu máy tính Deep Blue của IBM từng đánh bại đại kiện tướng Garry Kasparov vào năm 1997)

Một số ứng dụng của siêu máy tính

- Ảnh mô phỏng do siêu máy tính của Cơ quan Khí tượng và Hải dương Mỹ (NOAA) dựng ra bằng siêu máy tính của họ



Thị phần siêu máy tính



- IBM
- HP
- Cray Inc.
- Appro
- SGI
- Bull
- Dell
- NEC
- ManyCoreSoft
- Atipa
- Inspur
- NEC/HP
- IPE, Nvidia, Tyan
- Intel
- Clustervision/Supermicro
- Itautec
- Acer Group
- Xenon Systems
- Raytheon/Aspen Systems
- Eurotech
- Lenovo
- Self-made
- Dell/Sun/IBM
- Supermicro
- Dawning
- T-Platforms
- Adtech
- HP/WIPRO
- Fujitsu
- Oracle
- Hitachi
- NUDT
- NRCPCET
- Megware
- RSC Group

Thị phần siêu máy tính

Nước	Số lượng	Thị phần (%)	Rmax (GFlops)	Rpeak (Gflops)	Số nhân
Mỹ	251	50.2	89105145	129223344	8203546
Trung Quốc	72	14.4	12349895	22757659	1604920
Nhật	32	6.4	19448399	23362194	1389142
Anh	24	4.8	7260688	9393792	567152
Pháp	21	4.2	6413594	7887715	620348
Đức	19	3.8	10178296	12404695	875038
Canada	11	2.2	1858446	2429518	196744
Nga	8	1.6	1990634	3405775	179200
Ấn Độ	8	1.6	1167758	1757794	84148
Ý	7	1.4	2424814	3129931	222480
Australia	7	1.4	2196914	2689831	165828
Thụy Điển	6	1.2	993202	1295399	106080
Thụy Sĩ	4	0.8	978980	1254714	84416
Ba Lan	4	0.8	589563	947091	66690
Hàn Quốc	4	0.8	1014400	1291919	124536
Ả-rập Saudi	3	0.6	719300	1464275	116176
Đài Loan	3	0.6	356625	556484	38520
Na Uy	3	0.6	735400	873164	54400
Phần Lan	3	0.6	437691	527981	31344
Tây Ban Nha	2	0.4	740085	883092	39208
Brazil	2	0.4	465700	824638	48512
Áo	1	0.2	152900	182829	20776
Mexico	1	0.2	92282	116813	5616
Cộng hòa Slovak	1	0.2	76533	94274	3072
Đan Mạch	1	0.2	162098	183676	15672
Bỉ	1	0.2	152348	175718	8448
Israel	1	0.2	77696	161595	13788

HẾT CHƯƠNG 6