

RISC: Reduced Instruction Set Computer là kiến trúc tập lệnh lược giản

🡪về mục đích: được tạo ra để đơn giản hóa các lệnh riêng lẻ được được cung cấp cho máy tính hoàn thành nhiệm vụ.

Lquan đến CISC(Complex Instruction Set computer -máy tính với tập lệnh phức tạp) là một kiến trúc máy tính trong đó các lệnh máy tính được thiết kế để thực hiện nhiều chức năng khác nhau. Kiến trúc này có thể thực hiện các lệnh phức tạp chỉ bằng một lệnh duy nhất, giúp giảm thiểu số lượng lệnh số lượng lệnh cần thiết để thực hiện một tác vụ, tuy nhiên kiến trúc cx gây khó khăn cho các devs và không tối ưu cho các ứng dụng đa nhiệm

🡪Mối quan hệ giữa RISC và CISC: Trong RISC, tập lệnh được rút gọn và hầu hết các lệnh này rất nguyên thủy, trong khi ở CISC, tập lệnh rất lớn và có thể được sử dụng cho công việc hoặc tác vụ siêu phức tạp

Thời gian thực thi của máy tính RISC rất thấp, trong khi thời gian thực thi của máy tính CISC lại rất cao.

Ứng dụng của **pipelining** vào kiến trúc RISC:

* Pipelining(kỹ thuật ống dẫn) 🡪là một kỹ thuật mà trong đó các lệnh được thực thi theo kiểu chồng lấn lên nhau(overlap)🡪ví dụ

🡪ví dụ:

\_giả sử trong một căn phòng có nhiều người, ai cx cần giặt quần áo của mình🡪mọi người cần phải thực hiện 3 giai đoạn sau

* + Đặt quần áo bẩn vào máy và bấm nút
  + Khi máy giặt xog. Lấy ra phơi
  + Phơi xog cất vào tủ

.Không là pipelining: nếu một người hoàn tất tát cả các giai đoạn giặt quần áo nhưu trên thì người khác mới bắt đầu giặt(thực hiện lại 3 giai đoạn đó giống người thứ nhất) 🡪ko được gọi là pipelining

.Để nó là pipelining: Tuy nhiên, khi người đầu tiên hoàn thành xog giai đoạn 1 (tức là giai đoạn giặt trong máy) 🡪sang giai đoạn 2 là phơi thì máy giặt đã trống và người tiếp theo có thể sử dụng máy. Như vậy, người tiếp theo không cần phải chờ người đầu tiên hoàn thành cả 3 giai đoạn thì mới có thể bắt đầu, mà ngay khi người 1 xog bước đến giai đoạn thứ tiếp theo đó đã có thể bát đầu giai đoạn thứ nhất và cứ tiếp túc như vậy đến hết 🡪pipelining

* Vậy thì mối quan hệ giữa RISC và pipelining là gì:

🡪với mặt lợi của kỹ thuật này là giúp gia tăng hiệu suất cho máy và hầu hết các lệnh trong RISC đều là từ thanh ghi đến thanh ghi (register to register)

1. Tại sao CISC phát triển trước RISC:

Bộ nhớ hạn chế: Máy tính thời kỳ đầu có bộ nhớ rất hạn chế, vì vậy việc tạo nên 1 chương trình nhỏ gọn là rất quan trọng. CISC kết hợp nhiều hoạt động đơn giản hơn thành một hoạt động duy nhât 🡪 giảm thiểu số lượng câu lệnh 🡪 tối ưu bộ nhớ

Bộ xử lý chậm hơn: Lúc bấy giờ, bộ xử lý còn chậm chạp.Việc lấy các câu lệnh mất nhiều thời gian hơn so với việc thực thi chúng. CISC sẽ giải quyết bằng cách tạo ra số ít các câu lệnh phức tạp xử lí nhiều hoạt động nhằm mục đích giảm số lần lấy cần thiết.

🡪tóm lại là, CISC sẽ tối ưu phần cứng đơn giản bằng việc hạn chế tối đa số lượng lệnh cần thực thi

1. Đặc điểm chính của kiến trúc RISC:

* Bộ lệnh đơn giản🡪RISC sẽ sử dụng các tập lệnh đơn giản để giải mã và thực thi nhanh hơn🡪Mỗi câu lệnh chỉ thực hiên 1 hoạt động nhất định(load,store,…)
* Thực hiện trong một chu kì xung 🡪điều này dẫn đến thời gian xử lí nhanh hơn so với các lệnh phức tạp hơn
* Nhiều thanh ghi đa năng hơn🡪các vi xử lí RISC thường có số lượng lớn thanh ghi đa năng, nhiều thanh ghi này lưu trữ kết quả trung gian và toán hạng, giảm thiểu thời gian truy cập bộ nhớ
* Chế độ địa chỉ đơn giản🡪RISC sử dụng nhiều chế độ địa chỉ đơn giản như(register, immediate, base+offset) nhằm tối ưu hóa thực thi
* Thực thi theo ống dẫn (Pipelining)🡪RISC có thể thực hiện giai đoạn xử lí hiệu quả, trong đó nhiều lệnh đang ở các giai đoạn thực thi khác nhau đồng thời

PIPELINING:

* Pipelining(kỹ thuật ống dẫn) 🡪là một kỹ thuật mà trong đó các lệnh được thực thi theo kiểu chồng lấn lên nhau(overlap)🡪ví dụ

🡪ví dụ:

\_giả sử trong một căn phòng có nhiều người, ai cx cần giặt quần áo của mình🡪mọi người cần phải thực hiện 3 giai đoạn sau

* + Đặt quần áo bẩn vào máy và bấm nút
  + Khi máy giặt xog. Lấy ra phơi
  + Phơi xog khô cất vào tủ

.Không là pipelining: nếu một người hoàn tất tát cả các giai đoạn giặt quần áo nhưu trên thì người khác mới bắt đầu giặt(thực hiện lại 3 giai đoạn đó giống người thứ nhất) 🡪ko được gọi là pipelining

* CÁCH THỨC HOẠT ĐỘNG CỦA PIPELINING:

Pipeline là một kỹ thuật mà trong đó các lệnh được thực thi

theo kiểu chồng lấn lên nhau (overlap). 🡪ví dụ

* XUNG ĐỘT:

. Xung đột cấu trúc (Structural hazard): xung đột cấu trúc xảy ra khi có hai lệnh cùng truy xuất vào một tài nguyên phần cứng nào đó cùng một lúc.

. Xung đột dữ liệu (Data hazard): là khi một lệnh dự kiến không thể thực thi

trong đúng chu kỳ pipeline của nó do dữ liệu mà lệnh này cần vẫn chưa sẵn

sàng.

. Xung đột điều khiển (Control/Branch hazard): do lệnh nạp vào không phải là lệnh được cần.

* TỐI ƯU HÓA KỸ THUẬT PIPELINING:

. Delayed branch: trì hoãn nhánh tức là khi có nhánh chèn lệnh NOOP để trì hoãn đến khi một nhánh được thực hiện xog

. Delayed load: register bị khóa bởi vi xử lí, tiếp tục thực hiện câu lệnh cho đến khi yêu cầu thanh ghi,chờ đến khi load dữ liệu xong, sắp xếp lại các câu lệnh cho phép các công việc hữu ích

. Loop Unrolling(mở rộng vòng lặp): lặp ít lần hơn, tăng tính song song lệnh, cải thiện thanh ghi, bộ nhớ đêm,…

**PRESENTING**

1. Nhắc lại 1 tí về RISC 🡪 về mục đích: được tạo ra để đơn giản hóa các lệnh riêng lẻ được cung cấp cho máy tính hoàn thành nhiệm vụ.
2. Why CISC?

**! Bộ nhớ hạn chế**: Máy tính thời kỳ đầu có bộ nhớ rất hạn chế, vì vậy việc tạo nên 1 chương trình nhỏ gọn là rất quan trọng. CISC kết hợp nhiều hoạt động đơn giản hơn thành một hoạt động duy nhât 🡪 giảm thiểu số lượng câu lệnh 🡪 tối ưu bộ nhớ

**! Bộ xử lý chậm hơn**: Lúc bấy giờ, bộ xử lý còn chậm chạp.Việc lấy các câu lệnh mất nhiều thời gian hơn so với việc thực thi chúng. CISC sẽ giải quyết bằng cách tạo ra số ít các câu lệnh phức tạp xử lí nhiều hoạt động nhằm mục đích giảm số lần lấy cần thiết.

**🡪tóm lại là**, CISC sẽ tối ưu cho phần cứng đơn giản bằng cách hạn chế tối đa số lượng lệnh cần thiết cho việc thực thi

1. Mối quan hệ giữa RISC và CISC: Trong RISC, tập lệnh được rút gọn và hầu hết các lệnh này rất nguyên thủy, trong khi ở CISC, tập lệnh rất lớn và có thể được sử dụng cho công việc hoặc tác vụ siêu phức tạp

🡪Thời gian thực thi của máy tính RISC rất thấp, trong khi thời gian thực thi của máy tính CISC lại rất cao.

1. Một vài đặc điểm chính của RISC:
   * Bộ lệnh đơn giản🡪RISC sẽ sử dụng các tập lệnh đơn giản để giải mã và thực thi nhanh hơn🡪Mỗi câu lệnh chỉ thực hiên 1 hoạt động nhất định(load,store,…)
   * Ít kiểu dữ liệu hơn🡪Các lệnh RISC hoạt động trên một tập hạn chế kiểu dữ liệu (ví dụ: số nguyên, số thực). Điều này giúp thực thi hiệu quả.
   * Chế độ địa chỉ đơn giản🡪RISC sử dụng nhiều chế độ địa chỉ đơn giản như(register, immediate, base+offset) nhằm tối ưu hóa thực thi
   * Thực thi theo ống dẫn (Pipelining)🡪RISC có thể thực hiện giai đoạn xử lí hiệu quả, trong đó nhiều lệnh đang ở các giai đoạn thực thi khác nhau đồng thời
2. **Definition of Pipeline và cách thức hoạt động**: là một kỹ thuật mà trong đó các lệnh được thực thi theo kiểu chồng lấn lên nhau(overlap)🡪**ví dụ** máy giặt:

--> giả sử trong một căn phòng có nhiều người, ai cx cần giặt quần áo của mình🡪mọi người cần phải thực hiện 3 giai đoạn sau

* + Đặt quần áo bẩn vào máy và bấm nút
  + Khi máy giặt xog. Lấy ra phơi
  + Phơi xog khô cất vào tủ

.Không là pipelining: nếu một người hoàn tất tát cả các giai đoạn giặt quần áo nhưu trên thì người khác mới bắt đầu giặt(thực hiện lại 3 giai đoạn đó giống người thứ nhất) 🡪ko được gọi là pipelining

* + - Chú ý: ***Kỹ thuật pipeline không giúp giảm thời gian thực thi của từng lệnh riêng lẻ mà giúp giảm tổng thời gian thực thi của đoạn lệnh/chương trình chứa nhiều lệnh (từ đó giúp thời gian trung bình của mỗi lệnh giảm)***

1. Xung đột:

**XUNG ĐỘT:**

Xét một bộ xử lý với 8 lệnh cơ bản: load word (lw), store word (sw), add

(add), subtract (sub), AND (and), OR (or), set less than (slt), và nhảy với

điều kiện bằng (beq).

- Giả sử thời gian hoạt động các công đoạn như sau: 200 ps cho truy xuất

bộ nhớ, 200 ps cho tính toán của ALU, 100 ps cho thao tác đọc/ghi thanh

ghi

**. Xung đột cấu trúc** (Structural hazard): xung đột cấu trúc xảy ra khi có hai lệnh cùng truy xuất vào một tài nguyên phần cứng nào đó cùng một lúc.

🡪Giả sử rằng chúng ta có một bộ nhớ đơn duy nhất thay vì hai bộ nhớ lệnh và dữ

liệu rời rạc nhau. Nếu pipeline trong ví dụ ở hình trên có thêm lệnh thứ tư thì

trong chu kỳ pipeline từ 600 tới 800 khi lệnh thứ nhất thực hiện truy xuất bộ nhớ

lấy dữ liệu thì lệnh thứ tư sẽ thực hiện truy xuất bộ nhớ lấy lệnh. Do không có bộ

nhớ lệnh và dữ liệu riêng lẻ, trong trường hợp này sẽ có xung đột cấu trúc xảy ra. 🡨

**. Xung đột dữ liệu** (Data hazard): là khi một lệnh dự kiến không thể thực thi

trong đúng chu kỳ pipeline của nó do dữ liệu mà lệnh này cần vẫn chưa sẵn

sàng.

🡪Trong ví dụ trên, nếu áp dụng pipeline bình thường thì công đoạn ID của lệnh sub sẽ thực hiện cùng lúc với cộng đoạn EX của lệnh add. Trong công đoạn ID, lệnh sub sẽ cần đọc giá trị của thanh ghi $s0, trong khi đó giá trị mới của thanh ghi $s0 phải tới công đoạn WB của lệnh add mới sẵn sàng. Vì vậy, nếu thực hiện pipeline thông thường, trường hợp này sẽ xảy ra xung đột dữ liệu 🡨

. **Xung đột điều khiển** (Control/Branch hazard): do lệnh nạp vào không phải là lệnh được cần.

🡪Nếu áp dụng pipeline thông thường, tại chu kỳ thứ ba của pipeline, khi beq đang thực thi công đoạn ID thì lệnh lw sẽ được nạp vào. Nhưng nếu điều kiện bằng của lệnh beq xảy ra thì lệnh thực hiệp tiếp sau đó không phải là lw mà là lệnh được gán nhãn ‘label’, lúc này xảy ra xung đột điều khiển. 🡨

1. **TỐI ƯU HÓA KỸ THUẬT PIPELINING**:

. **Delayed branch**: trì hoãn nhánh tức là khi có nhánh chèn lệnh NOOP để trì hoãn đến khi một nhánh được thực hiện xog

. **Delayed load**: register bị khóa bởi vi xử lí, tiếp tục thực hiện câu lệnh cho đến khi yêu cầu thanh ghi,chờ đến khi load dữ liệu xong, sắp xếp lại các câu lệnh cho phép các công việc hữu ích

. **Loop Unrolling**(mở rộng vòng lặp): lặp ít lần hơn, tăng tính song song lệnh, cải thiện thanh ghi, bộ nhớ đêm,…