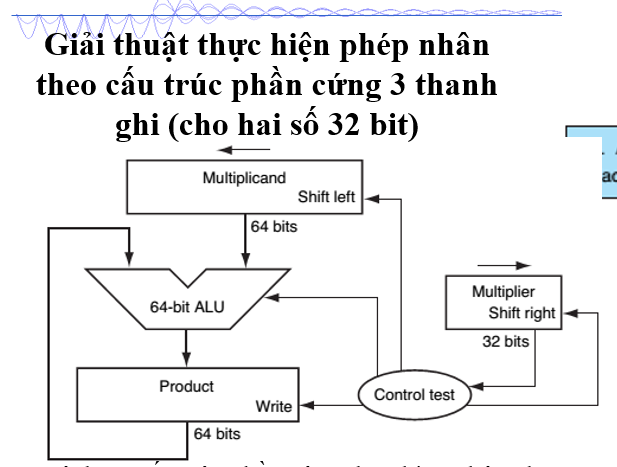
Cách 1: 2 thanh ghi 64 bit



Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

Cách 2: 2 thanh ghi 32 bit, 1 thanh ghi 64 bit ( 50 x 23, 50 = 101000 và 23 = 010011 )

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

**Divisor: số chia**

**Dividend: số bị chia**

**Quotient: thương số**

**Remainder: số dư**Ảnh có chứa bàn

Mô tả được tạo tự động

chia sẽ hơn nhân 1 bước nên sẽ hơn 1 bước lặp

Sau khi phép chia thực hiện xong:

Hi chứa phần dư

Lo chứa thương số

Để xử lý cho các số có dấu và số không dấu, MIPS có 2 lệnh: phép chia có dấu (div), và phép chia không dấu (divu).

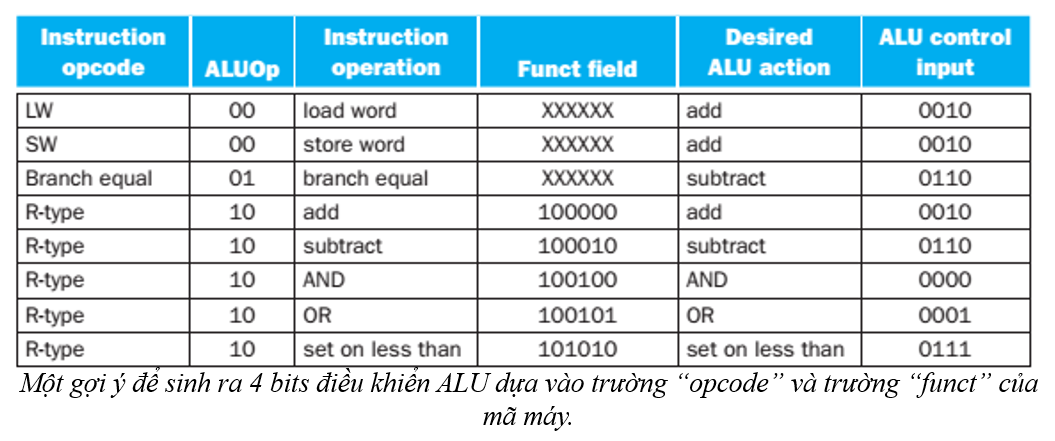
Ảnh có chứa bàn

Mô tả được tạo tự động

Với các lệnh **load word** và **store word**, ALU sử dụng chức năng **‘add’** để tính toán địa chỉ của bộ nhớ

Với các lệnh thuộc **nhóm logic và số học**, ALU thực hiện 1 trong 5 chức năng (***AND, OR, subtract, add,*** *và* ***set on less than***), tùy thuộc vào giá trị của trường funct (6 bits) trong mã máy lệnh.

Với lệnh **nhảy nếu bằng**, ALU thực hiện chức năng **‘subtract’** để xem điều khiện bằng có đúng không.



Ảnh có chứa bàn

Mô tả được tạo tự động

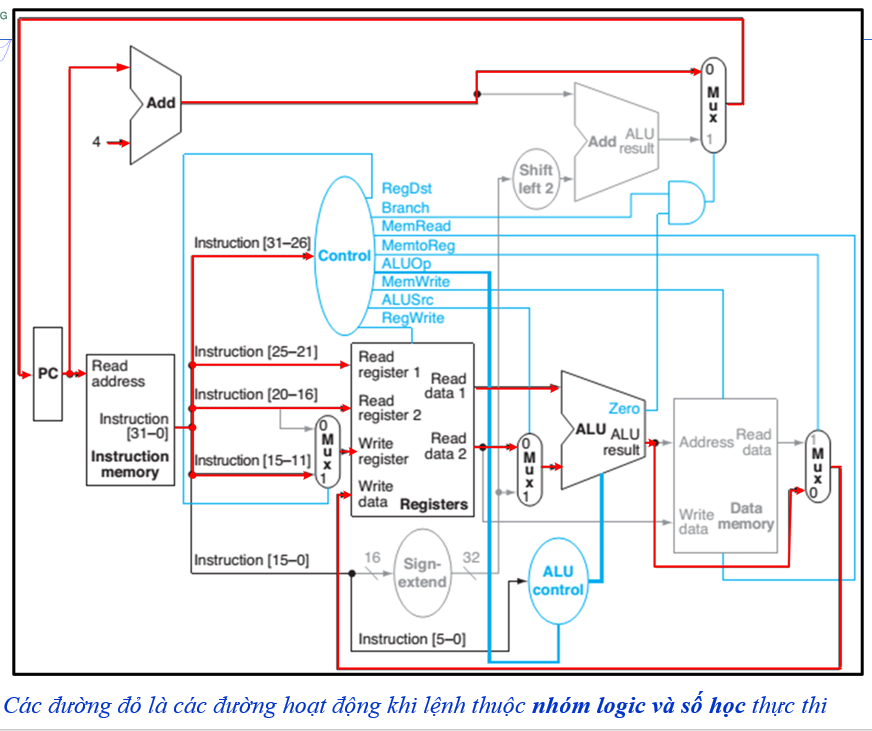
Ảnh có chứa bàn

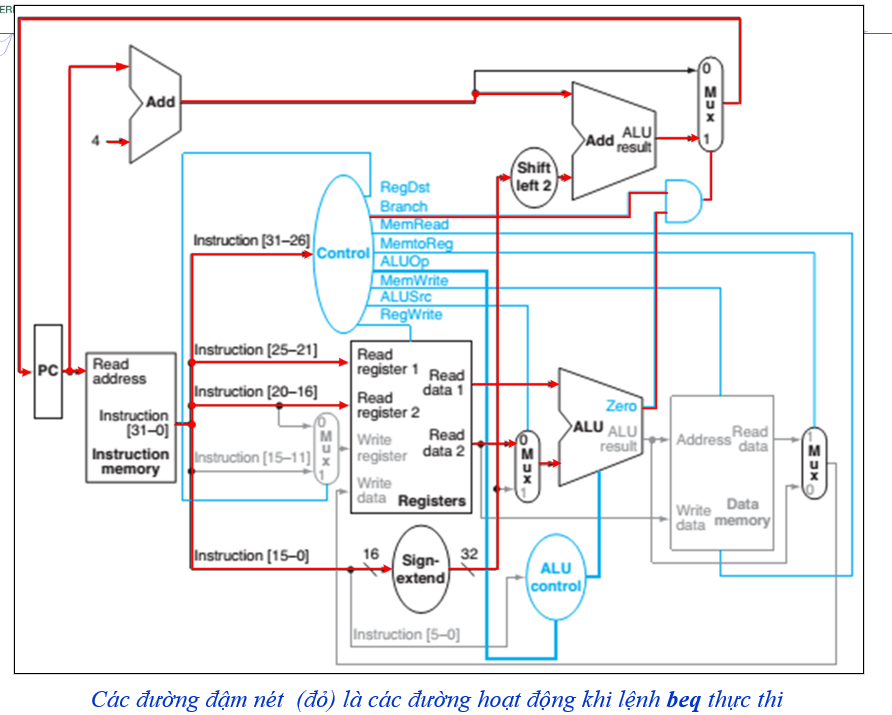
Mô tả được tạo tự động

Ảnh có chứa bàn

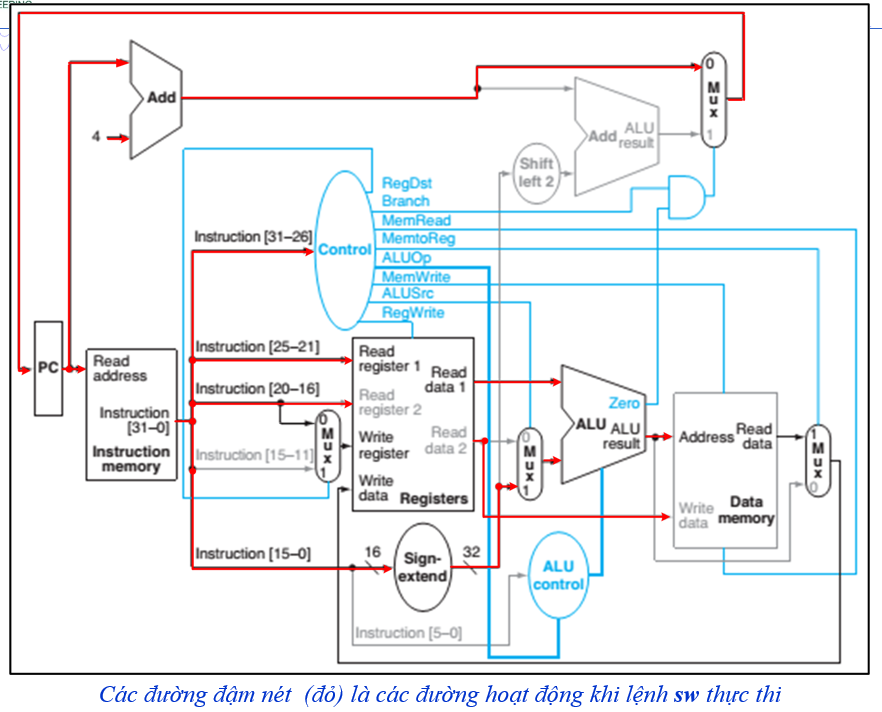
Mô tả được tạo tự động

**Hiện thực bộ xử lý đơn chu kỳ** (Single-cycle implementation hay single clock cycle implementation): là cách hiện thực sao cho bộ xử lý đáp ứng thực thi mỗi câu lệnh chỉ trong 1 chu kỳ xung clock 🡺 đòi hỏi chu kỳ xung clock phải bằng thời gian của lệnh dài nhất.





Lệnh SW và LW giống nhau.



**PIPELINE**

Nạp lệnh từ bộ nhớ − **IF**

Giải mã lệnh và đọc các thanh ghi − **ID**

Thực thi − **EX**

Truy xuất bộ nhớ − **MEM**

Ghi kết quả vào thanh ghi – **WB**

**Trong trường hợp lý tưởng, pipeline sẽ tăng tốc so với không pipeline với số lần đúng bằng số tầng của pipeline.**

**Trong thực tế:** Các công đoạn thực tế không bằng nhau, việc áp dụng pipeline phải chọn công đoạn dài nhất để làm một chu kỳ pipeline**🡪 Trong thực tế, pipeline sẽ tăng tốc so với không pipeline với số lần nhỏ hơn số tầng của pipeline.**

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)

**Xung đột** là trạng thái mà lệnh tiếp theo không thể thực thi trong chu kỳ pipeline ngay sau đó (hoặc thực thi nhưng sẽ cho ra kết quả sai), thường do một trong ba nguyên nhân sau:

* **Xung đột cấu trúc (Structural hazard):** làkhi một lệnh dự kiến không thể thực thi trong đúng chu kỳ pipeline của nó do phần cứng cần không thể hỗ trợ. Nói cách khác, 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):** là khi một lệnh dự kiến không thể thực thi trong đúng chu kỳ pipeline của nó do lệnh nạp vào không phải là lệnh được cần. Xung đột này xảy ra trong trường hợp luồng thực thi chứa các lệnh nhảy.

đơn chu ký, không pipeline

đa chu kỳ, có pipeline