KIẾN TRÚC HAVARD

Kiến trúc Havard là 1 kiến trúc máy tính với các đường dẫn tín hiệu và bộ nhớ riêng biệt cho các lệnh và dữ liệu. Nó trái ngược với kiến trúc Von Neumann nơi các lệnh chương trình và dữ liệu chia sẻ cùng một bộ nhớ và đường dẫn (đường truyền)

Thuật ngữ này có nguồn gốc từ máy tính Havard Mark I, máy tính này lưu trữ các lệnh trên các băng đục lỗ (24 bit) và dữ liệu trong các bộ đếm cơ điện. Những máy tính đầu tiên này có bộ lưu trữ dữ liệu nằm hoàn toàn trong bộ xử lý trung tâm và không cho phép truy cập vào bộ lưu trữ lệnh dưới dạng dữ liệu. Các chương trình cần được tải bởi 1 người điều hành, bộ xử lý không thể tự khởi tạo.

Các bộ vi xử lý hiện đại đến với người dùng là các máy Von Neuman, với mã chương trình được lưu trong cùng bộ nhớ chính với dữ liệu. Vì lý do hiệu suất, bên trong và phần lớn không hiện thị với người dùng, hầu hết các thiết kế đều có bộ nhớ đệm riêng biệt cho các lệnh và dữ liệu, với các đường dẫn riêng biệt đến bộ xử lý cho từng thiết kế. Đây là 1 dạng của kiến trúc Havard đã được sửa đổi.

CHI TIẾT BỘ NHỚ

Trong kiến trúc Havard, không cần thiết phải làm cho các bộ nhớ có cùng đặc điểm chung. Trong 1 số hệ thống, các lệnh cho các tác vụ được lập trình trước có thể được lưu trữ trong bộ nhớ chỉ đọc trong khi bộ nhớ dữ liệu thường yêu cầu bộ nhớ đọc - ghi. Trong 1 số hệ thống, có nhiều bộ nhớ lệnh hơn bộ nhớ dữ liệu nên địa chỉ lệnh rộng hơn địa chỉ dữ liệu.

Đối ngược với kiến trúc Von Neumann

Trong 1 hệ thống có kiến trúc Von Neuman thuần túy, các lệnh và dữ liệu được lưu trữ trong cùng 1 bộ nhớ, do đó các lệnh được tìm trên cùng 1 đường dẫn được dùng để tìm dữ liệu. Điều này có nghĩa là CPU không thể đồng thời đọc ~~hoặc ghi~~ 1 lệnh ~~hoặc~~ và đọc hoặc ghi dữ liệu từ hoặc vào bộ nhớ. Trong 1 máy tính sử dụng kiến trúc Havard, CPU có thể vừa đọc lệnh, vừa thực hiện truy cập bộ nhớ dữ liệu cùng 1 lúc ngay cả khi không có bộ nhớ đệm. 1 máy tính kiến trúc Havard có thể nhanh hơn với độ phức tạp của 1 mạch nhất định ~~do~~ bởi việc tìm nạp lệnh và truy cập dữ liệu không ~~đụng độ~~ ảnh hưởng đến nhau trên một đường dẫn bộ nhớ duy nhất.

Ngoài ra 1 máy kiến trúc Havard có mã và không gian địa chỉ dữ liệu riêng biệt: địa chỉ lệnh zero không giống với địa chỉ dữ liệu zero. Địa chỉ lệnh zero có thể xác định giá trị 24 bit, trong khi địa chỉ dữ liệu zero có thể chỉ ra 1 byte 8 bit cái mà không phải là 1 phần của giá trị 24 bit.

Đối ngược với kiến trúc Havard đã sửa đổi

Máy kiến trúc Havard đã sửa đổi rất giống với máy kiến trúc Havard nhưng nó nới lỏng sự tách biệt chặt chẽ giữa lệnh và dữ liệu trong khi vẫn cho phép CPU truy cập đồng thời 2 (hoặc nhiều hơn) bộ nhớ bus. Việc sửa đổi phổ biến nhất bao gồm bộ nhớ đệm và lệnh và dữ liệu riêng biệt được hỗ trợ bởi 1 không gian địa chỉ chung. Trong khi CPU thực thi từ bộ nhớ đệm, nó hoạt động như 1 máy Havard thuần túy. Khi truy cập bộ nhớ hỗ trợ, nó hoạt động giống như 1 máy Von Neumann (nơi mã có thể được di chuyển xung quanh ~~như dữ~~ giống như dữ liệu, đây là 1 kĩ thuật mạnh mẽ). Sự sửa đổi này sống rải rác trong các bộ xử lý hiện đại, chẳng hạn như kiến trúc ARM, POWER ISA và bộ xử lý x86. Đối

thì nó thường gọi là cấu trúc dài là kiến trúc Harvard, nhưng thực ra nó là kiến trúc Harvard đã được sửa đổi.

1 sửa đổi phổ cung cấp 1 đường truyền giữa bộ nhớ lệnh (chẳng hạn như ROM hoặc bộ nhớ flash) và CPU cho phép các từ từ bộ nhớ lệnh được coi là dữ liệu chỉ đọc. Kỹ thuật này được sử dụng trong 1 số vi điều khiển, bao gồm cả Atmel AVR. Nó cho phép dữ liệu không đổi, chẳng hạn như chuỗi văn bản hoặc bảng hằng, được truy cập mà không cần phải sao chép trước vào bộ nhớ dữ liệu, bảo toàn bộ nhớ dữ liệu khả biến (và nguồn điện) cho các biến đọc/ghi. Các lệnh ngôn ngữ máy đặc biệt được cung cấp để đọc dữ liệu từ bộ nhớ lệnh, hoặc bộ nhớ lệnh có thể được truy cập bằng giao diện ngoại vi.

TỐC ĐỘ

Trong những năm gần đây, tốc độ của CPU đã tăng gấp nhiều lần so với tốc độ truy cập của bộ nhớ chính. Cần phải cẩn thận giảm số lần truy cập bộ nhớ chính để duy trì hiệu suất. Ví dụ mọi lệnh chạy trong CPU đều yêu cầu quyền truy cập vào bộ nhớ, máy tính sẽ không thể được gì khi tốc độ CPU tăng lên - một vấn đề được gọi là sàng buộc bộ nhớ.

Có thể tạo ra bộ nhớ cực nhanh, nhưng điều này chỉ thực tế đối với số lượng nhỏ bộ nhớ do chi phí, điện năng và tín hiệu truyền tín hiệu. Giải pháp là cung cấp 1 số lượng nhỏ bộ nhớ rất nhanh được gọi là bộ nhớ đệm CPU chứa dữ liệu được truy cập gần đây. Miễn là dữ liệu mà CPU cần nằm trong bộ nhớ đệm thì hiệu suất sẽ cao hơn nhiều so với khi CPU phải lấy dữ liệu từ bộ nhớ chính.

THIẾT KẾ TRONG SO VỚI THIẾT KẾ BÊN NGOÀI

Các thiết kế chip CPU hiệu suất cao hiện đại kết hợp với các khía cạnh của cả kiến trúc Harvard và Von Neumann. Đặc biệt, phiên bản "bộ đệm phân chia" của kiến trúc Harvard đã sửa đổi là rất phổ biến. Bộ nhớ đệm CPU được chia thành bộ nhớ đệm lệnh và bộ nhớ đệm dữ liệu. Kiến trúc Harvard được sử dụng khi CPU truy cập vào bộ đệm. Tuy nhiên, trong trường hợp bộ đệm bị bỏ sót, dữ liệu được truy xuất từ bộ nhớ chính, nhưng không được chứng thực chia thành các phiên dữ liệu và lệnh riêng biệt, mặc dù nó cũng có thể ~~đồng thời~~ điều khiển bộ nhớ riêng được sử dụng để truy cập đồng thời vào RAM, ROM và (hầu hết) bộ nhớ flash.

Do đó, trong khi kiến trúc Von Neumann có thể hiện thì trong một số ngữ cảnh, chẳng hạn như khi dữ liệu và mã đi qua cùng 1 bộ điều khiển bộ nhớ, thì việc triển khai phần cứng thực tế của kiến trúc Harvard đối với truy cập bộ nhớ đệm và ít nhất là 1 số truy cập vào bộ nhớ chính.

Ngoài ra, CPU thường có bộ ~~nhớ~~ đệm ghi cho phép CPU tiến hành sau khi ghi vào các vùng không được ~~lưu trữ~~ lưu trong bộ đệm. Bản chất Von Neumann của bộ nhớ khi đó được hiển thị khi các lệnh được viết dưới dạng dữ liệu bởi CPU và phần mềm phải đảm bảo rằng bộ đệm (dữ liệu và lệnh) và bộ đệm ghi được đồng bộ hóa trước khi cố gắng thực hiện các lệnh vừa viết đó.

CÔNG DỤNG HIỆN ĐẠI CỦA KIẾN TRÚC HARVARD

Lợi thế chính của kiến trúc Harvard thuần túy — khả năng truy cập đồng thời vào nhiều hệ thống bộ nhớ — đã bị giảm bớt bởi các bộ xử lý Harvard đã sửa đổi sử dụng hệ thống bộ nhớ đệm CPU hiện đại. Máy kiến trúc Harvard thường đối thuần túy

được sử dụng ~~trong~~ hầu hết trong các ứng dụng mà sự cân bằng, như chi phí và tiết kiệm điện thì việc bỏ qua bộ nhớ đệm, vượt trội hơn các bất lợi lập trình do có một và không gian địa chỉ dữ liệu riêng biệt.

Các bộ xử lý tín hiệu kỹ thuật số (DSP) thường thực hiện các thuật toán xử lý âm thanh hoặc video nhỏ, ~~hoạt~~ đòi hỏi tốc độ cao. Chúng tránh bộ nhớ đệm vì cách ~~mà~~ xử lý của chúng phải mức kỹ để tái tạo.

Các bộ vi ~~xử lý~~ điều khiển có đặc điểm là 1 không nhớ chương trình (bộ nhớ flash) và bộ nhớ dữ liệu (SRAM) đồng thời tận dụng lợi thế của kiến trúc Harvard để tăng tốc độ xử lý bằng lệnh và truy cập dữ liệu đồng thời.

Ngay cả trong những trường hợp này, người ta thường sử dụng các lệnh đặc biệt để truy cập bộ nhớ chương trình như thể nó là dữ liệu cho các ~~lệnh~~ chỉ đọc hoặc để lập trình lại; những bộ xử lý đó là bằng bộ xử lý kiến trúc Harvard đã sửa đổi.