HỆ THỐNG MÁY TÍNH VÀ NGÔN NGỮ C

CHƯƠNG 4

MÔ HÌNH VON NEUMANN VÀ KIẾN TRÚC TẬP LỆNH LC-3

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

CHƯƠNG 4: MÔ HÌNH VON NEUMANN VÀ KIẾN TRÚC TẬP LỆNH LC-3

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

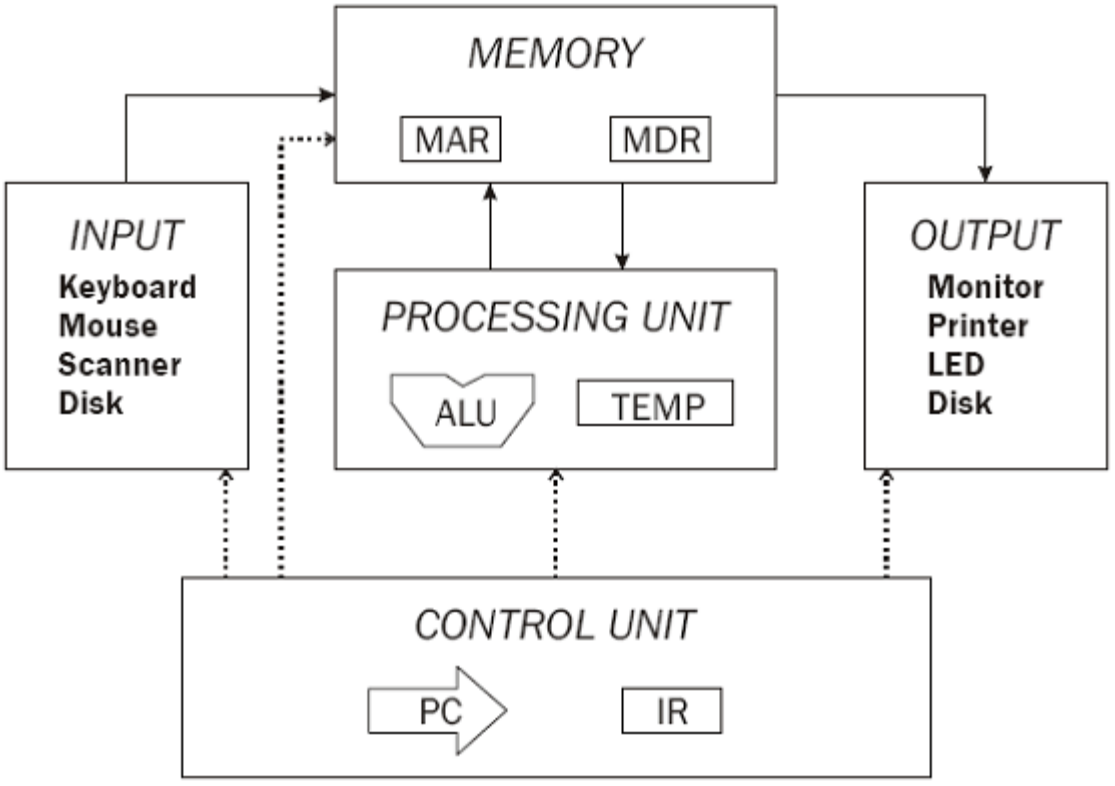
Vào năm 1946, John von Neumann đã đưa ra một mô hình máy tính cơ bản để xử lý các chương trình máy tính gồm năm bộ phận cơ bản:

- bộ nhớ (memory)
- đơn vị xử lý (processing unit)
- thiết bị nhập (input)
- thiết bị xuất (output)
- đơn vị điều khiển (control unit).

Chương trình máy tính được chứa trong bộ nhớ của máy tính. Việc điều khiển thứ tự các lệnh cần thực hiện sẽ do đơn vị điều khiển đảm trách.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản



Hình 4.1 Sơ đồ khối tổng quát mô hình von Neumann

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.1 Bộ nhớ (Memory)

Tổng quát, với số bit địa chỉ là k, chúng ta có thể biểu diễn được 2^k phần tử nhớ.

Với kiến trúc tập lệnh của máy tính LC-3, chúng ta có không gian địa chỉ là 2^{16}, và mỗi phần tử dài 16 bit.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.1 Bộ nhớ (Memory)

Có hai thao tác truy xuất bộ nhớ là đọc và ghi.

Đọc thông tin của một ô nhớ:

- Đặt địa chỉ của ô nhớ đó vào thanh ghi địa chỉ bộ nhớ MAR (Memory Address Register)
- Tín hiệu Read tích cực
- Sau một thời gian, thông tin từ ô nhớ có địa chỉ trên sẽ được đặt vào thanh ghi dữ liệu bộ nhớ MDR (Memory Data Register).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.1 Bộ nhớ (Memory)

Lưu một giá trị vào một ô nhớ:

- Ghi địa chỉ của ô nhớ đó vào thanh ghi MAR và giá trị cần lưu vào thanh ghi MDR.
- Tín hiệu Write Enable tích cực.
- Khi đó, thông tin đang ở trong thanh ghi MDR sẽ được ghi vào ô nhớ có địa chỉ trong thanh ghi MAR.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.1 Bộ nhớ (Memory)

Hình 4.2 Bộ nhớ 16 phần tử, mỗi phần tử 8 bit

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.2 Đơn vị xử lý (Processing Unit)

Đơn vị xử lý là bộ phận thực sự trong máy tính xử lý thông tin. (chia, căn bậc hai, ….)

Theo mô hình von Neumann bộ phận xử lý chính là đơn vị số học luận lý ALU (Arithmetic Logic Unit) vì nó có thể thực hiện các phép tính số học như cộng, trừ, và các thao tác logic cơ bản như AND, OR, và NOT.

Các thao tác mà ALU của LC-3 có thể thực hiện là ADD, AND, và NOT.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.2 Đơn vị xử lý (Processing Unit)

Kích thước của các toán hạng được ALU xử lý thường được xem như là chiều dài từ máy của máy tính.

Mỗi toán hạng được xem là một từ.

Trong LC-3, ALU xử lý toán hạng 16 bit. Chúng ta nói LC-3 có chiều dài từ 16 bit. (32 bit như Intel Pentium 4 hoặc 64 bit như SUN SPARC-V9 và Intel Core i3.)

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.2 Đơn vị xử lý (Processing Unit)

Ngoài ra, để thực hiện tốt thao tác trong thời gian ngắn nhất, trong đơn vị xử lý còn có một bộ nhớ tạm, đó là tập các thanh ghi, mỗi thanh ghi có cấu trúc như trong mục 3.4.3.

Kích thước của thanh ghi luôn bằng với kích thước của toán hạng đầu vào của ALU, có nghĩa là mỗi thanh ghi chứa một từ máy.

LC-3 có tám thanh ghi (R0, R1, …, R7), mỗi thanh ghi dài 16 bit. Cấp ISA của SPARC-V9 có 32 thanh ghi (R0, R1, …, R31), mỗi thanh ghi dài 64 bit.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.3 Xuất và nhập

Để một máy tính xử lý thông tin, thông tin phải được đưa vào trong máy tính. Để sử dụng được kết quả đã được xử lý, các kết quả này phải được thể hiển bằng một cách nào đó ra bên ngoài máy tính. Các thiết bị làm các việc như vậy gọi là các thiết bị xuất nhập, còn được gọi là các thiết bị ngoại vi.

Trong LC-3, chúng ta có hai thiết bị xuất nhập, đó là bàn phím và màn hình.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.4 Đơn vị điều khiển (Control Unit)

Đơn vị điều khiển cũng như nhạc trưởng của một dàn nhạc, nó có nhiệm vụ làm tất cả các bộ phận khác làm việc với nhau.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.1 Các thành phần cơ bản

4.1.4 Đơn vị điều khiển

Để theo dõi lệnh nào đang được thực thi, đơn vị điều khiển có thanh ghi lệnh IR (instruction register) để chứa lệnh đó.

Để theo dõi lệnh cần được thực thi kế tiếp, đơn vị điều khiển có một thanh ghi chứa địa chỉ của lệnh kế đó, PC (program counter),.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3

Các đầu mũi tên tô đặc ký hiệu cho các phần tử dữ liệu chạy theo đường truyền tương ứng.

Các đầu mũi tên không tô đặc ký hiệu cho các tín hiệu điều khiển dùng để điều khiển các phần tử khác hoạt động.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3.

Các bộ phận trong mô hình von Neumann của LC-3 là:

1. Bộ nhớ (Memory) gồm các phần tử lưu trữ, cùng với thanh ghi MAR chỉ tới ô nhớ riêng biệt, và thanh ghi MDR giữ nội dung của ô nhớ trong quá trình ghi/đọc bộ nhớ.

    - Thanh ghi MAR dài 16 bit phản ảnh không gian địa chỉ bộ nhớ của LC-3 là 2^{16} ô nhớ.
    - Thanh ghi MDR dài 16 bit, cho biết thông tin trong mỗi ô nhớ là 16 bit.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3

2. Xuất/ Nhập (Input/Output) :Gồm bàn phím và màn hình.

Để thao tác với bàn phím, ta có hai thanh ghi, thanh ghi dữ liệu KBDR (Keyboard Data Register) giữ mã ASCII của các phím đã được nhấn, và thanh ghi trạng thái KBSR (Keyboard Status Register) lưu thông tin về trạng thái của phím được ấn.

Màn hình cũng cần hai thanh ghi để làm việc, thanh ghi DDR (Display Data Register) giữ mã ASCII của cái cần hiển thị, và thanh ghi DSR (Display Status Register) giữ thông tin về trạng thái hoạt động của màn hình.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3

3.Đơn vị xử lý (Processing unit)

Gồm đơn vị số học luận lý ALU và tám thanh ghi (R0, …, R7) để lưu các giá trị tạm thời cần cho quá trình tham khảo, tính toán trong tương lai. ALU của LC-3 có thể thực hiện một phép tính số học (cộng) và hai thao tác luận lý (AND và bù 1).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

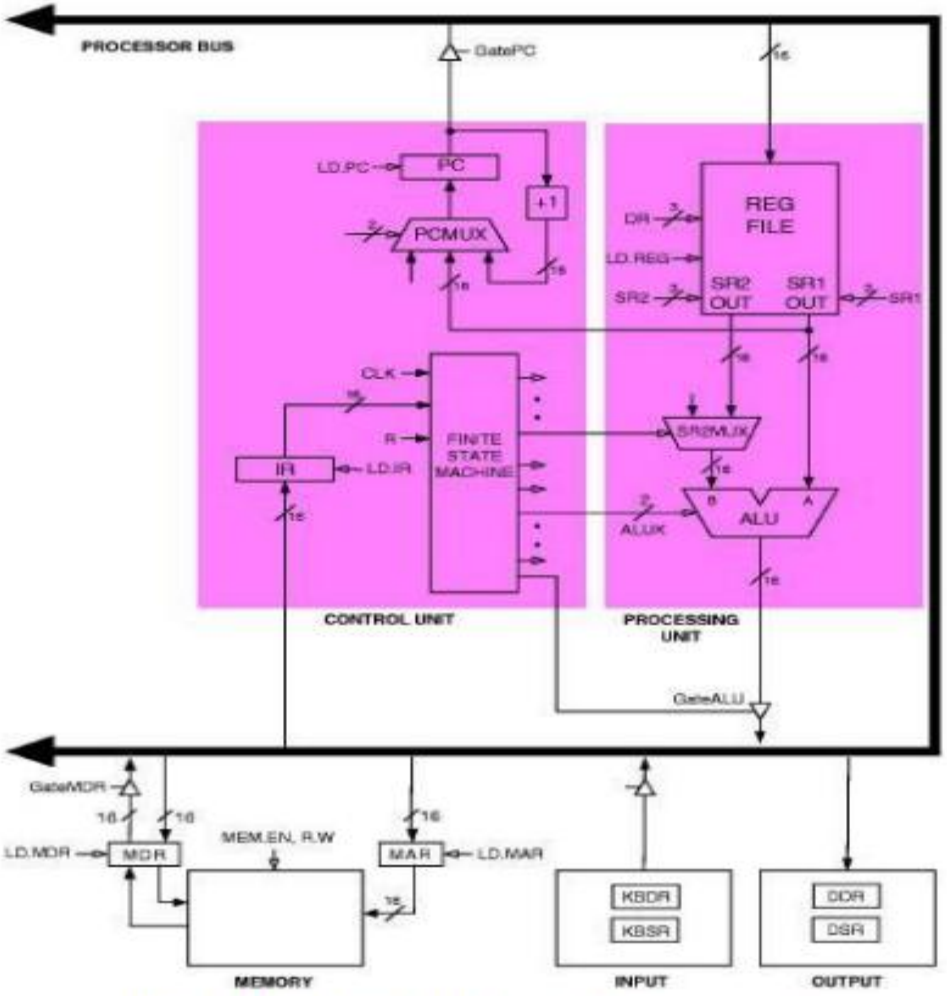
4.2 Một ví dụ về mô hình von Neumann: LC-3

4. Đơn vị điều khiển (Control unit) gồm tất cả các phần tử cần thiết để quản lý quá trình đang được máy tính xử lý. Cấu trúc quan trọng nhất là máy trạng thái hữu hạn (Finite state machine), điều khiển tất cả các hoạt động. Nó hoạt động theo từng bước, từ chu kỳ xung clock này qua chu kỳ xung clock khác(CLK)

   Thanh ghi IR (instruction register) cũng là một đầu vào của máy trạng thái hữu hạn, để xác định các thao tác cần thực hiện trong quá trình thực thi lệnh LC-3 đang có trong thanh ghi IR. Thanh ghi PC (program counter) cũng là một phần của đơn vị điều khiển, nó theo dõi lệnh kế cần được thực thi sau khi lệnh hiện thời hoàn thành.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3



Hình 4.3 LC-3 là một ví dụ cho mô hình von Neumann

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.2 Một ví dụ về mô hình von Neumann: LC-3

Chú ý, tất cả đầu ra từ máy trạng thái hữu hạn trong hình 4.3 đều là các tín hiệu điều khiển, nên chúng đều có đầu mũi tên rỗng.

Ví dụ, một trong các đầu ra là 2 bit ALUK, dùng để quy định thao tác mà ALU cần thực hiện (add, and, và not) trong chu kỳ xung clock hiện hành. Đầu ra khác là GateALU, để quyết định việc xuất dữ liệu ra processor bus.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

Một vấn đề quan trọng trong mô hình von Neumann là quá trình xử lý chương trình và dữ liệu được lưu giữ dưới dạng chuỗi các bit trong bộ nhớ máy tính.

Quá trình gồm nhiều bước, mỗi bước có chức năng riêng mà ta cần tìm hiểu.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.1 Lệnh

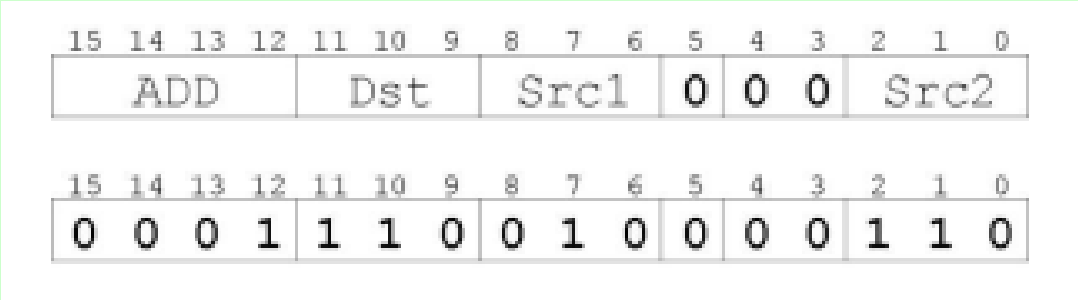
Đơn vị cơ bản nhất của quá trình xử lý của máy tính là lệnh. Lệnh gồm hai phần, mã lệnh (opcode) và toán hạng (operand). Với LC-3, mỗi lệnh gồm 16 bit, được đánh số từ trái qua phải từ bit[15] tới bit[0]. Bit[15:12] chứa opcode. Điều này có nghĩa là có tổng cộng 2^4 mã lệnh khác nhau. Các bit từ bit[11:0] được dùng để xác định toán hạng.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.1 Lệnh

Ví dụ 4.1: Lệnh ADD (cộng) có ba toán hạng gồm hai toán hạng nguồn (dữ liệu từ đó được cộng) và một toán hạng đích (giữ tổng sau khi phép cộng được thực thi). Vì ISA LC-3 có tám thanh ghi nên có ba bit cần dùng để mã cho một thanh ghi. Lệnh cộng ADD có dạng như sau:

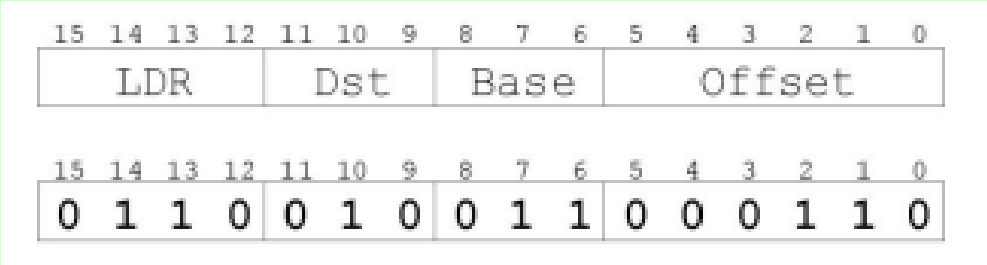


CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.1 Lệnh

Ví dụ 4.2: Lệnh LDR (LD xuất phát từ Load) sẽ vào ô nhớ được xác định, đọc dữ liệu và lưu nó vào thanh ghi. Lệnh này có hai toán hạng là giá trị đọc được từ ô nhớ và thanh ghi đích. Ký tự R trong LDR viết tắt từ register, cho biết cơ chế kiểu địa chỉ (addressing mode) xác định toán hạng là ô nhớ lưu dữ liệu, cụ thể là Base + offset. Lệnh này có dạng như sau:



CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

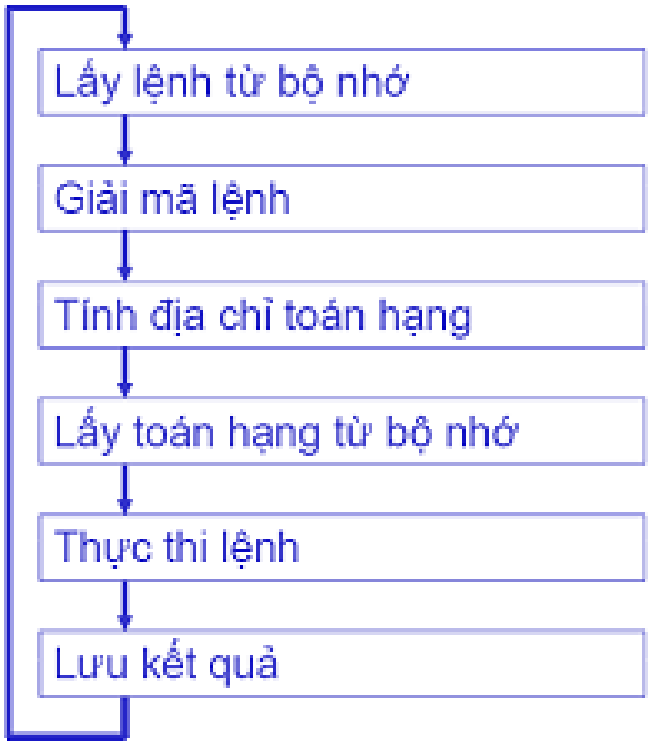
4.3.2 Chu kỳ lệnh

Dưới sự điều phối của đơn vị điều khiển, các lệnh được xử lý một cách hệ thống theo từng bước. Chuỗi các bước thực hiện lệnh được gọi là chu kỳ lệnh (instruction cycle). Mỗi bước được xem như một pha (phase).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh



Hình 4.4 Sáu pha của chu kỳ lệnh

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh

1. Lấy lệnh từ bộ nhớ (Fetch)

FETCH sẽ gồm các thao tác sau:

Bước 1: Nạp nội dung của thanh ghi PC vào thanh ghi MAR, đồng thời tăng PC

Bước 2: Yêu cầu bộ nhớ để lấy lệnh đặt vào thanh ghi MDR

Bước 3: Nạp vào thanh ghi IR nội dung của thanh ghi MDR

Mỗi bước thao tác như vậy chiếm một hay nhiều chu kỳ máy (hay chu kỳ xung clock của CPU), như bước 1 và 3 mỗi bước chiếm một chu kỳ máy, trong khi bước 2 có thể chiếm hơn một chu kỳ máy tùy theo thời gian truy xuất bộ nhớ.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh

2. Giải mã lệnh (Decode)

Pha này khảo sát lệnh để đưa ra các yêu cầu cho vi kiến trúc thực hiện.

Trong LC-3, bộ giải mã 4 ra 16 xác định mã lệnh nào trong 16 mã lệnh cần được xử lý.

Đầu vào là bốn bit mã lệnh IR[15:12]. Đường đầu ra tích cực tương ứng với mã lệnh ở đầu vào. Tùy thuộc vào đầu ra nào của bộ giải mã được tích cực.

IR[11:0] xác định cái cần phải làm để thực hiện lệnh đó.

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh

3. Tính địa chỉ toán hạng (Evaluate address)

Pha này sẽ tính địa chỉ của ô nhớ cần thiết để xử lý lệnh. Như lệnh LDR trong ví dụ 4.2 sẽ lấy một trị từ một ô nhớ và nạp vào thanh ghi. Trong ví dụ này, địa chỉ ô nhớ sẽ được tính từ trị 6 cộng với nội dung thanh ghi R3. Thao tác tính toán này được thực hiện trong pha này.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh

4. Lấy toán hạng (Fetch Operands)

Pha này thực hiện việc lấy các toán hạng cần thiết để xử lý lệnh. Trong ví dụ 4.2 về lệnh LDR, pha này gồm hai bước: nạp thanh ghi MAR giá trị địa chỉ tính toán được từ pha Tính địa chỉ toán hạng, và đọc bộ nhớ để có toán hạng nguồn được đặt trong thanh ghi MDR.

Trong ví dụ 4.1 về lệnh ADD, pha này cần hai toán hạng từ hai thanh ghi R2 và R6. Với nhiều bộ vi xử lý hiện đại, việc lấy toán hạng từ thanh ghi có thể được thực hiện cùng lúc khi lệnh được giải mã để tăng tốc quá trình xử lý lệnh.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.3 Quá trình xử lý lệnh

4.3.2 Chu kỳ lệnh

5. Thực thi lệnh (Execute)

Pha này thực thi lệnh sau khi đã có đủ tất cả toán hạng và mã lệnh. Như trong ví dụ về lệnh ADD, pha này chỉ gồm bước đơn giản là thực thi việc cộng trong ALU.

6. Lưu kết quả (Store result)

Đây là sau cùng của quá trình thực thi lệnh. Kết quả của lệnh sẽ được ghi vào đích đã được xác định.

Một khi pha thứ sau này được hoàn thành, đơn vị điều khiển bắt đầu một chu kỳ lệnh mới, từ đầu pha pha Lấy lệnh.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.4 Thay đổi quá trình xử lý lệnh

Bình thường một chương trình máy tính được thực hiện theo trình tự, nghĩa là lệnh đầu tiên được thực thi, sau đó tới lệnh thứ hai, rồi thứ ba, ….

Có một nhóm lệnh đặc biệt gọi là lệnh điều khiển, nó có thể thay đổi trình tự thực thi lệnh.

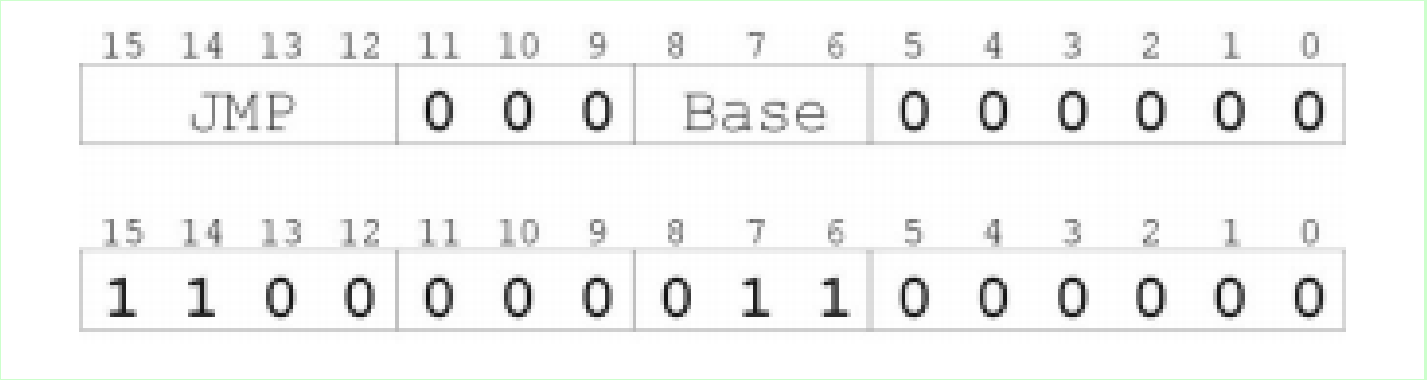
Như chúng ta biết, mỗi chu kỳ lệnh bắt đầu bằng việc nạp thanh ghi PC vào thanh ghi MAR. Như vậy, nếu chúng ta muốn thay đổi trình tự thực thi lệnh, chúng ta phải thay đổi thanh ghi PC trong khoảng giữa thời gian nó được tăng lên (trong pha Lấy lệnh của một lệnh) và sự bắt đầu của pha Lấy lệnh của lệnh kế.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.4 Thay đổi quá trình xử lý lệnh

Các lệnh điều khiển thực hiện chức năng đó bằng việc nạp thanh ghi PC trong pha Thực thi lệnh, việc này sẽ xóa trị đã được tăng trong thanh ghi PC trong pha Lấy lệnh trước đó.

Ví dụ 4.3: Lệnh JMP của ISA LC-3 có định dạng như sau. Giả sử lệnh này đang được chứa trong bộ nhớ ở địa chỉ x36A2.



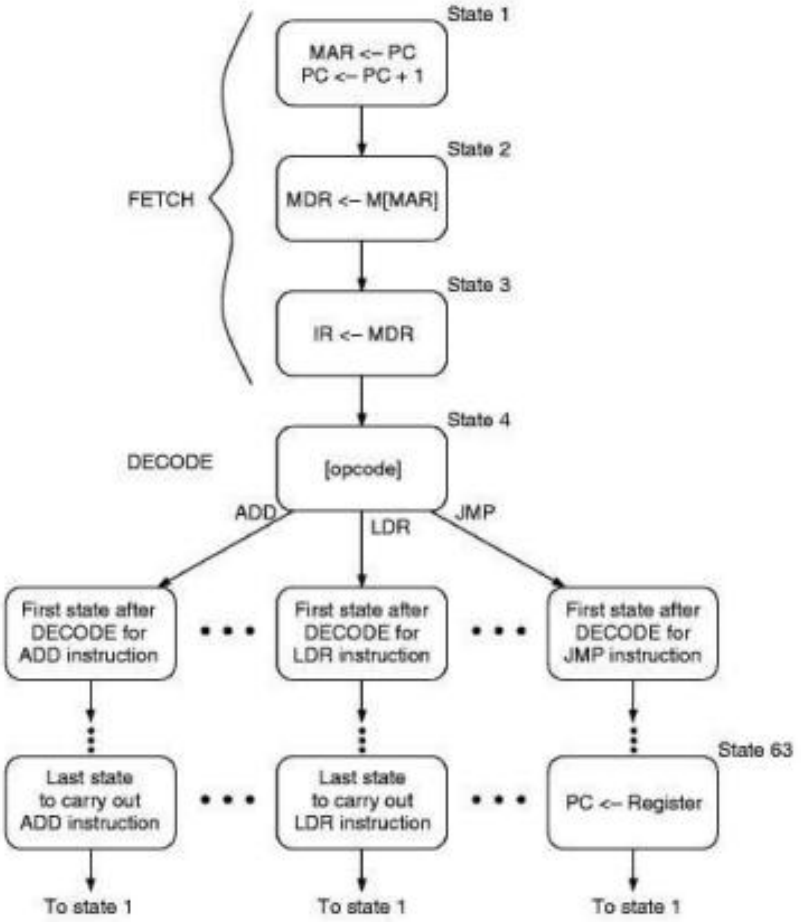
4.4 Thay đổi quá trình xử lý lệnh

Giả sử thanh ghi R3 đang chứa x5446, trong pha Thực thi lệnh, thanh ghi PC sẽ được nạp trị x5446 này.

Do đó trong chu kỳ lệnh kế tiếp, lệnh được xử lý là lệnh ở địa chỉ x5446 chứ không phải lệnh ở địa chỉ x36A3.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.4 Thay đổi quá trình xử lý lệnh



Hình 4.5 Sơ đồ trạng thái thu gọn của LC-3

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

Kiến trúc tập lệnh (ISA – Instruction Set Architecture) xác định tất cả thông tin về máy tính mà phần mềm cần phải nhận biết. Kiến trúc tập lệnh (ISA) xác định mọi thứ trong máy tính mà người lập trình sử dụng khi viết chương trình bằng ngôn ngữ máy.

(ISA xác định tổ chức bộ nhớ, tập thanh ghi, và tập lệnh gồm mã lệnh, loại dữ liệu, và các kiểu định vị địa chỉ.)

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.1 Tổ chức bộ nhớ

Bộ nhớ của LC-3 có dung lượng 2^{16} (65536) ô nhớ, mỗi ô có chiều dài 16 bit. Tuy nhiên, không phải tất cả 65536 ô nhớ đều được sử dụng, vì có một số vùng nhớ được dùng để lưu các thông tin hệ thống như bảng các vector ngắt, biến hệ thống, …. Vì đơn vị lưu trữ chuẩn được LC-3 xử lý là 16 bit, nên chúng ta gọi 16 bit là một từ (word), và do đó chúng ta nói LC-3 định vị theo từ.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.2 Thanh ghi

LC-3 thường tốn hơn một chu kỳ xung clock để lấy dữ liệu từ bộ nhớ, nên nó cũng cung cấp các thanh ghi để chứa dữ liệu tạm thời mà có thể được truy xuất chỉ trong một chu kỳ xung clock.

Các thanh ghi có tính chất giống như ô nhớ, tức nó được sử dụng để chứa thông tin mà có thể được truy tìm sau đó. Trong LC-3, mỗi thanh ghi dài một từ, tức 16 bit, và có 8 thanh ghi đa dụng. Mỗi thanh ghi có một chỉ định riêng, nên cần dùng 3 bit để mã cho số hiệu của một thanh ghi, đó là các thanh ghi R0, R1, …, R7.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.3 Tập lệnh

Một lệnh được tạo từ hai thứ, mã thao tác (opcode) là cái mà lệnh bắt máy tính thực thi, và toán hạng (operands) là cái mà máy tính cần để thực thi lệnh.

Tập lệnh của một ISA được định nghĩa bằng tập các mã thao tác, kiểu dữ liệu và các kiểu định vị để xác định vị trí của toán hạng.

Trong ví dụ 4.12 trên, ta có lệnh R6 = R2 + R6, thì kiểu định vị của toán hạng là thanh ghi vì các toán hạng đều là thanh ghi.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.4 Mã thao tác

Kiến trúc tập lệnh của LC-3 có 15 lệnh, mỗi lệnh được chỉ định mã thao tác riêng.

Mã thao tác được quy định trong bốn bit [15:12] của lệnh, nên sẽ có 16 mã thao tác khác nhau. Tuy nhiên, trong thực tế ISA LC-3 chỉ sử dụng 15 mã thao tác. Mã 1101 chưa được quy định mã thao tác, nó được để dành cho các nhu cầu cần thiết trong tương lai.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.4 Mã thao tác

Có ba loại lệnh khác nhau, tức có ba loại mã thao tác:

- Các lệnh thi hành xử lý thông tin, như lệnh ADD.

- Các lệnh chuyển dữ liệu chuyển thông tin qua lại giữa bộ nhớ và các thanh ghi, giữa các thanh ghi với nhau, giữa các thiết bị xuất nhập, như lệnh LDR.

- Các lệnh điều khiển thay đổi trình tự các lệnh sẽ được thực thi, như lệnh JMP.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.5 Các kiểu dữ liệu

Một kiểu dữ liệu là một sự miêu tả về thông tin để ISA có các mã thao tác thực thi với miêu tả đó. Có rất nhiều cách để biểu diễn một thông tin trên máy tính, ví dụ với một số nguyên ta có thể viết số ở hệ thập phân, hệ bát phân hay hệ thập lục phân. Hoặc với số thực, ta có thể dùng số thực dấu chấm cố định hay số thực dấu chấm động.

Trong kiến trúc tập lệnh LC-3, kiểu dữ liệu duy nhất được sử dụng là dạng số nguyên bù 2.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.6 Các kiểu định vị địa chỉ

Một kiểu định vị địa chỉ là một cơ chế để xác định toán hạng ở đâu. Tổng quát, một toán hạng có thể được tìm ở một trong ba chỗ: trong bộ nhớ, trong một thanh ghi, hoặc là một phần của lệnh. Nếu nó là một phần của lệnh, ta gọi nó là toán hạng tức thời.

LC-3 sử dụng năm kiểu định vị địa chỉ: tức thời, thanh ghi, và ba kiểu định vị địa chỉ bộ nhớ là PC-relative, gián tiếp và Base+offset.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.5 Khái niệm ISA LC-3

4.5.7 Các mã điều kiện

Hầu như tất cả các ISA đều cho phép quá trình thực hiện lệnh được thay đổi tùy thuộc vào kết quả được tạo ra trước đó.

LC-3 có ba thanh ghi một bit được đặt (set to 1) hay xóa (xóa về 0) mỗi khi một trong tám thanh ghi đa dụng được ghi trị. Ba thanh ghi một bit này được gọi là các thanh ghi trạng thái N, Z, và P

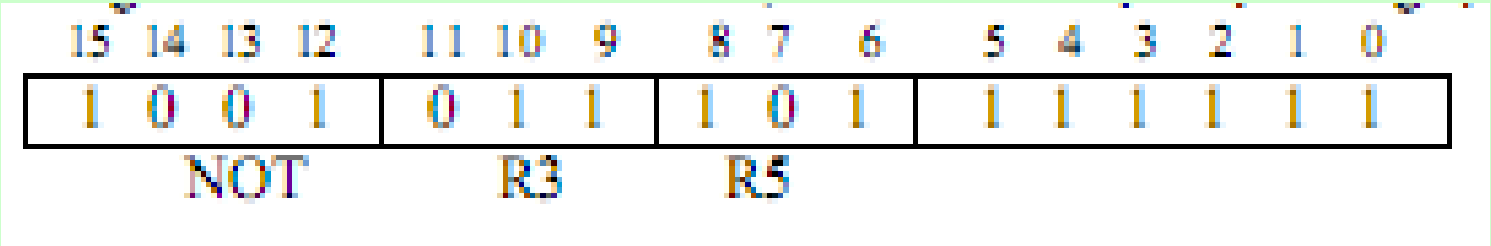
CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD+	0001				DR			SR1			0	00		SR2		
ADD+	0001				DR			SR1			1	imm5				
AND+	0101				DR			SR1			0	00		SR2		
AND+	0101				DR			SR1			1	imm5				
BR	0000				n	z	p	PCoffset9								
JMP	1100				000			BaseR			000000					
JSR	0100				1	PCoffset11										
JSRR	0100				0	00		BaseR			000000					
LD+	0010				DR			PCoffset9								
LDI+	1010				DR			PCoffset9								
LDR+	0110				DR			BaseR			offset6					
LEA+	1110				DR			PCoffset9								
NOT+	1001				DR			SR			111111					
RET	1100				000			111			000000					
RTI	1000				000000000000											
ST	0011				SR			PCoffset9								
STI	1011				SR			PCoffset9								
STR	0111				SR			BaseR			offset6					
TRAP	1111				0000			trapvect8								
reserved	1101															

Hình 4.3 Toàn bộ tập lệnh LC-3

4.6 Nhóm lệnh thi hành

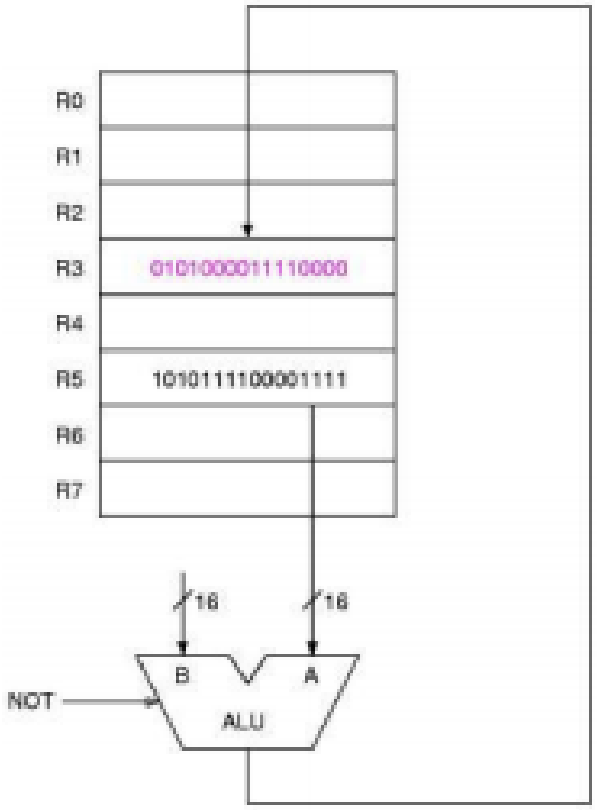
Ví dụ 4.4: Nếu R5 đang chứa 0101000011110000, thì sau khi thực hiện xong lệnh sau:



thanh ghi R3 sẽ lưu 1010111100001111.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.6 Nhóm lệnh thi hành



Hình 4.4 Đường truyền dữ liệu tương ứng với sự thực thi lệnh NOT R3, R5

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.6 Nhóm lệnh thi hành

Ví dụ 4.5: Nếu R4 chứa trị 6 và R5 chứa trị -18, sau khi lệnh dưới được thực thi

thì thanh ghi R1 chứa trị -12.

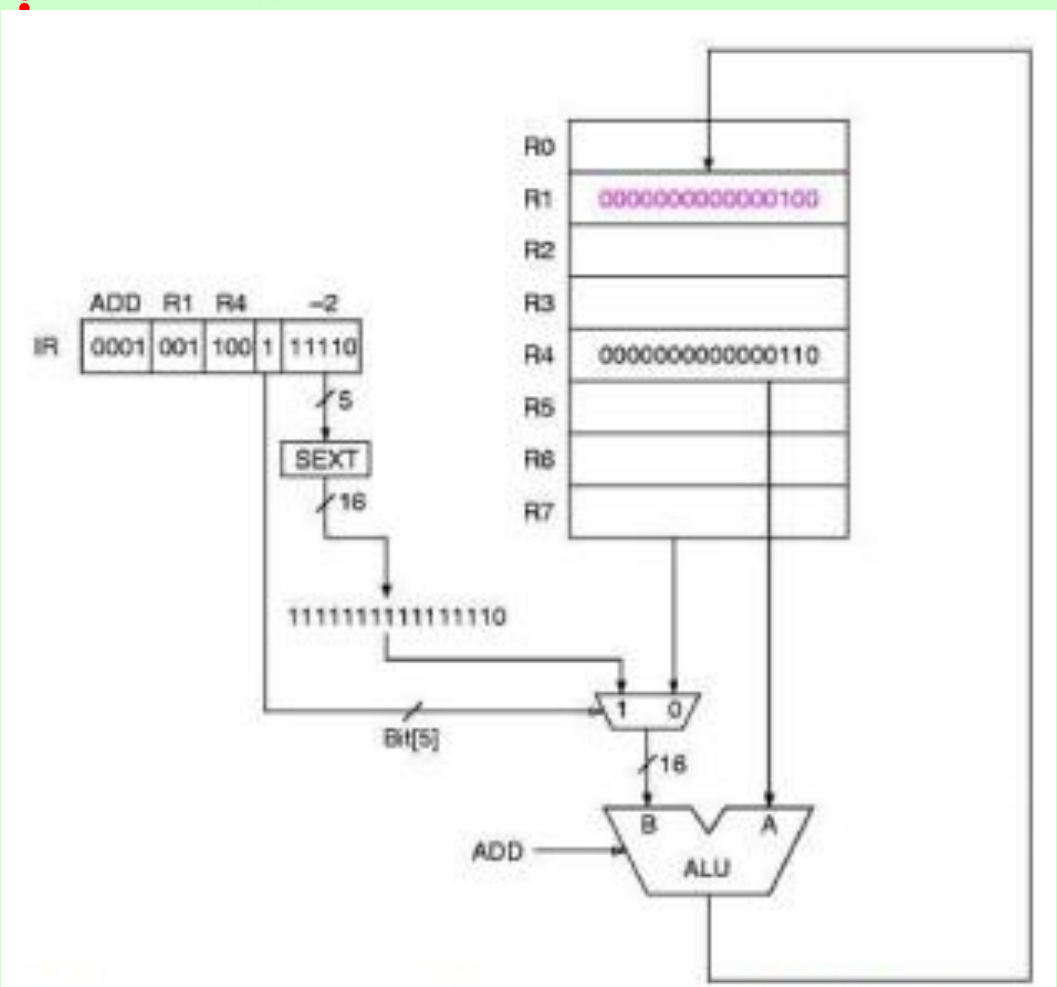
4.6 Nhóm lệnh thi hành

Nếu bit [5] là 1, toán hạng nguồn thứ hai được chứa ngay bên trong lệnh, và có được bằng phép mỡ rộng dấu 5 bit [4:0] thành 16 bit trước khi thực hiện phép ADD hoặc AND. Hình 4.5 trình bày các phần quan trọng của đường truyền dữ liệu để thực hiện lệnh ADD R1, R4, #-2.

Cũng cần lưu ý là vùng trị tức thời này chỉ có 5 bit ở dạng bù 2, nên nó chỉ có thể chứa trị trong khoảng -16..15 mà thôi.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.6 Nhóm lệnh thi hành

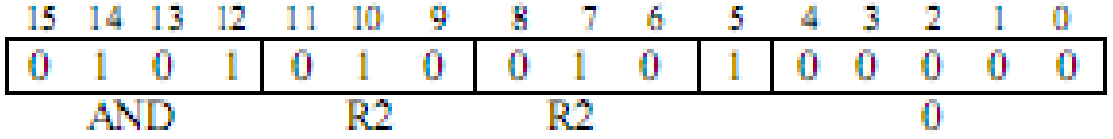


Hình 4.5 Đường truyền dữ liệu với sự thực thi lệnh ADD R1, R4, #-2

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.6 Nhóm lệnh thi hành

Ví dụ 4.6 : Lệnh **AND R2, R2, 0** sẽ xóa thanh ghi R2 về 0.



Ví dụ 4.7 : Lệnh **ADD R6, R6, 1** sẽ tăng thanh ghi R6 lên 1.

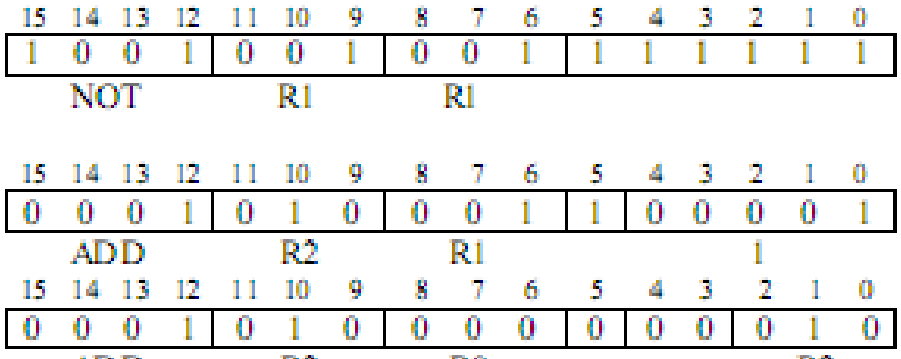
CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.6 Nhóm lệnh thi hành

Ví dụ 4.8: ba lệnh dưới đây thực hiện việc tính hiệu hai trị đang nằm trong hai thanh ghi R0 và R1.

Hai lệnh đầu (NOT R1, R1 và ADD R2, R1, 1) sẽ tính bù 2 của trị đang có trong thanh ghi R1.

Lệnh ADD R2, R0, R2 tính hiệu giữa hai trị ban đầu trong R0 và R1 theo yêu cầu.



CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

Nhóm lệnh chuyển dữ liệu thực hiện việc sao chép thông tin qua lại giữa các thanh ghi đa dụng và bộ nhớ, giữa các thanh ghi và các thiết bị xuất nhập.

Quá trình chép thông tin từ bộ nhớ vào thanh ghi được gọi là nạp (load), và quá trình chép thông tin từ thanh ghi vào bộ nhớ gọi là lưu (store) . Trong cả hai trường hợp, thông tin trong toán hạng nguồn không đổi, còn thông tin cũ trong toán hạng đích đã bị ghi đè bởi thông tin mới sau quá trình chép.

LC-3 có bảy lệnh thực hiện việc sao chép dữ liệu: LD, LDR, LDI, LEA, ST, STR, và STI.

Định dạng chung cho các lệnh này là

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

Các lệnh chuyển dữ liệu cần hai toán hạng, một nguồn và một đích. Toán hạng nguồn là nơi dữ liệu cần được chép, còn toán hạng đích là là nơi cần chép thông tin vào.

Trong định dạng trên, các bit [15:12] xác định mã thao tác opcode, các bit [11:9] quy định một trong hai toán hạng, là thanh ghi có mã theo quy ước. Các bit [8:0] chứa các bit tạo địa chỉ (Address generation bits), mã hóa thông tin để tính ra địa chỉ 16 bit của toán hạng thứ hai.

LC-3 có bốn kiểu định vị địa chỉ: định vị tương đối từ PC (PC-Relative mode), định vị gián tiếp (Indirect mode), định vị base + offset (base+offset mode), và định vị tức thời (immediate mode).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.1 PC-relative mode

Lệnh LD (opcode = 0010) và ST (opcode = 0011) dùng kiểu định vị tương đối PC. Các bit [8:0] của lệnh xác định một độ dời (offset) tính từ thanh ghi PC. Địa chỉ bộ nhớ được tính từ tổng của 16 bit địa chỉ sau phép mở rộng dấu từ các bit [8:0] và thanh ghi PC đã được tăng 1 sau pha FETCH. Với lệnh LD, ô nhớ tương ứng với địa chỉ bộ nhớ đã tính được đọc, và ghi vào thanh ghi được xác định bởi các bit [11:9] của lệnh.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.1 PC-relative mode

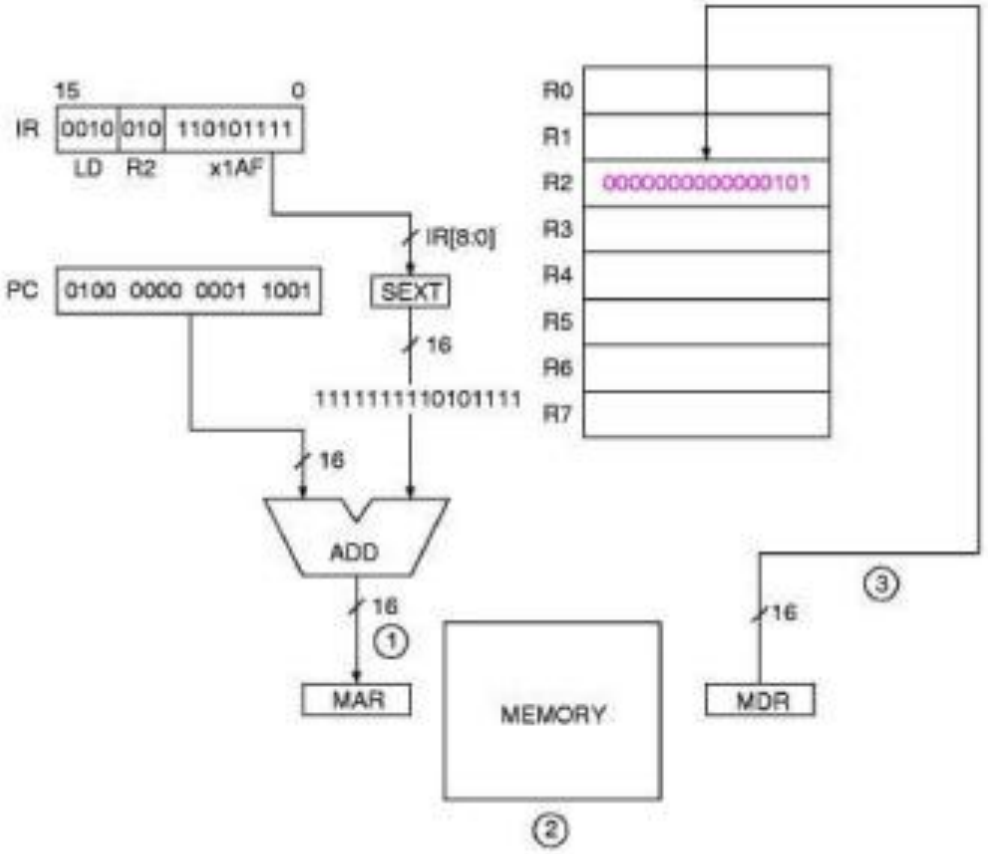
Ví dụ 4.9: Nếu lệnh

nằm ở ô nhớ có địa chỉ x4018, thì nội dung ô nhớ có địa chỉ x4019 + SEXT (x1AF) = x4019 + xFFAF = x3FC8 (bỏ bit 1 do vượt quá chiều dài 16 bit) sẽ được đọc và nạp vào thanh ghi R2.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.1 PC-relative mode



Hình 4.6 Đường truyền dữ liệu tương ứng lệnh LD R2, x1AF

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.2 Indirect mode

Lệnh LDI (opcode = 1010) và lệnh STI (opcode = 1011) sử dụng kiểu định vị địa chỉ gián tiếp. Với kiểu định vị này, địa chỉ đầu tiên được tạo ra cũng tương tự như với lệnh LD và ST.

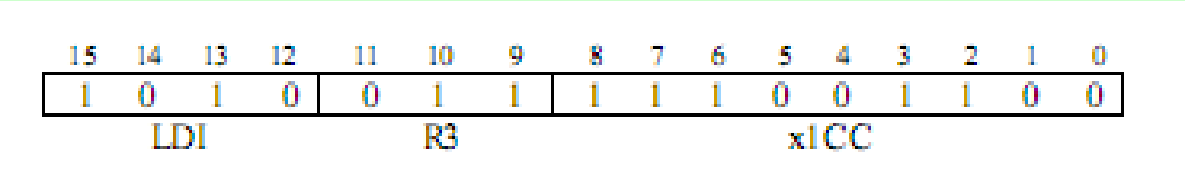
Tuy nhiên, thay vì đây địa chỉ của toán hạng chứa dữ liệu cần được nạp hay lưu, thì toán hạng này lại chứa địa chỉ của toán hạng là dữ liệu cần được nạp hay lưu. Vì phải qua trung gian như vậy nên ta mới có kiểu định vị địa chỉ gián tiếp. Với kiểu định vị này, ta thấy toán hạng chứa dữ liệu cần làm việc có thể ở bất cứ đâu trong bộ nhớ 64K, chứ không bị giới hạn trong tầm 9 bit [8:0] như trong trường hợp lệnh LD và ST. Thanh ghi đích cho LDI và thanh ghi nguồn trong STI cũng được xác định bằng các bit [11:9] trong lệnh.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.2 Indirect mode

Ví dụ 4.10: Nếu lệnh

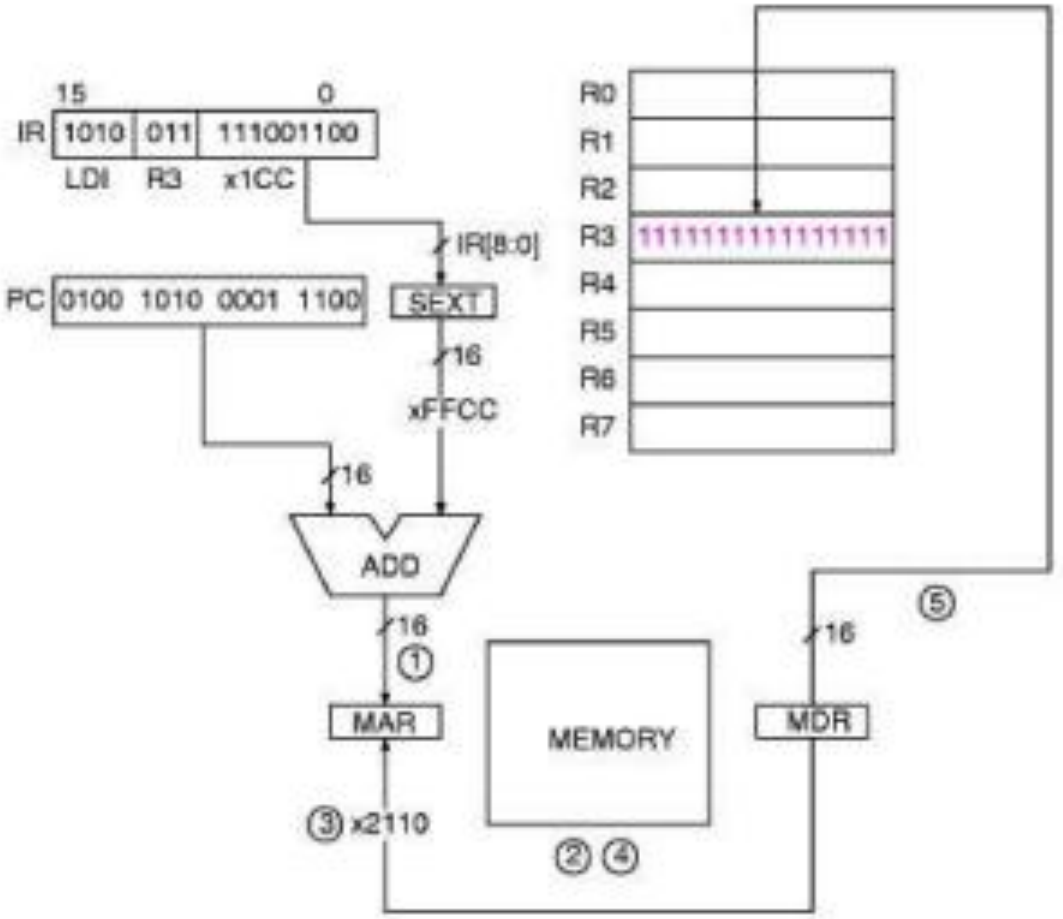


ở địa chỉ x4A1B, và chứa đựng của x49E8 là x2110, việc thực thi lệnh này sẽ lấy dữ liệu từ ô nhớ có địa chỉ x2110 nạp vào thanh ghi R3.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.2 Indirect mode



Hình 4.7 Đường truyền dữ liệu cho việc thực thi lệnh LDI R3, x1CC

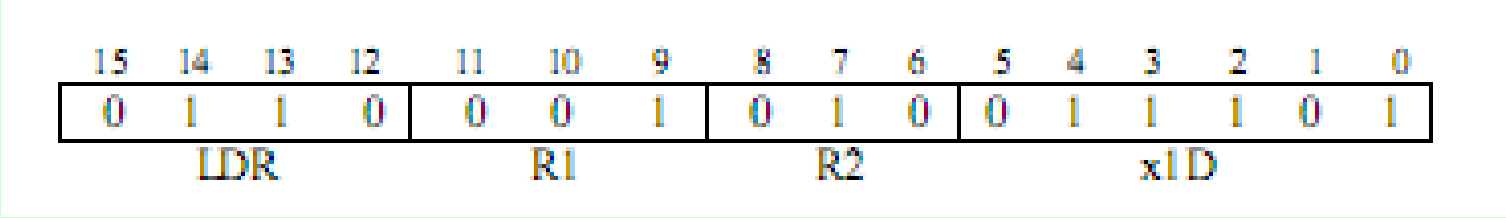
CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.3 Base+offset mode

Lệnh LDR (opcode = 0110) và STR (opcode = 0111) dùng kiểu định vị Base+offset. Kiểu định vị này xác định địa chỉ toán hạng trong bộ nhớ 64K bằng cách lấy 6 bit offset được mở rộng dấu IR[5:0] cộng với thanh ghi nền trong vùng IR[8:6].

Ví dụ 4.11: Nếu R2 đang chứa trị 16 bit x2345, lệnh

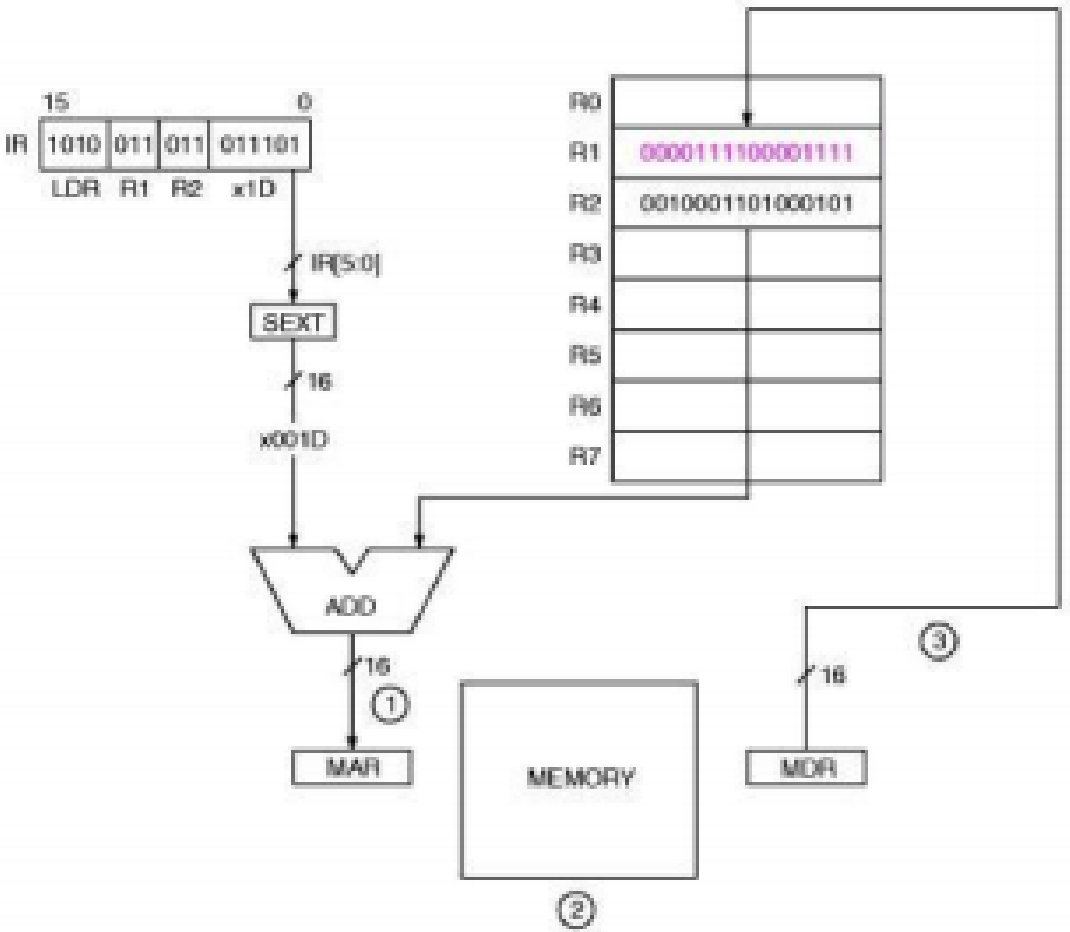


nạp R1 nội dung của ô nhớ x2362 (x2345 + x1D).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.3 Base+offset mode



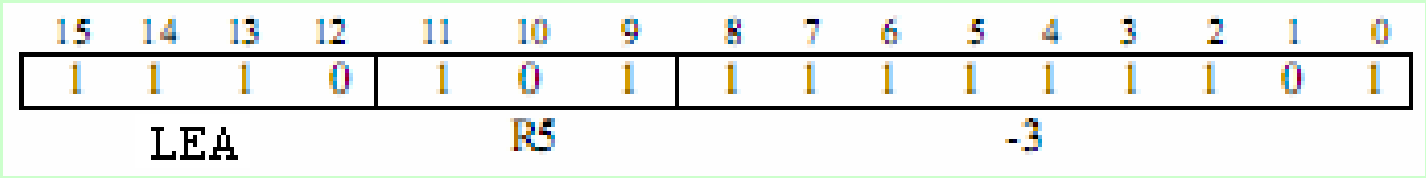
Hình 4.8 Đường truyền dữ liệu khi thực hiện lệnh LDR R1, R2, x1D

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

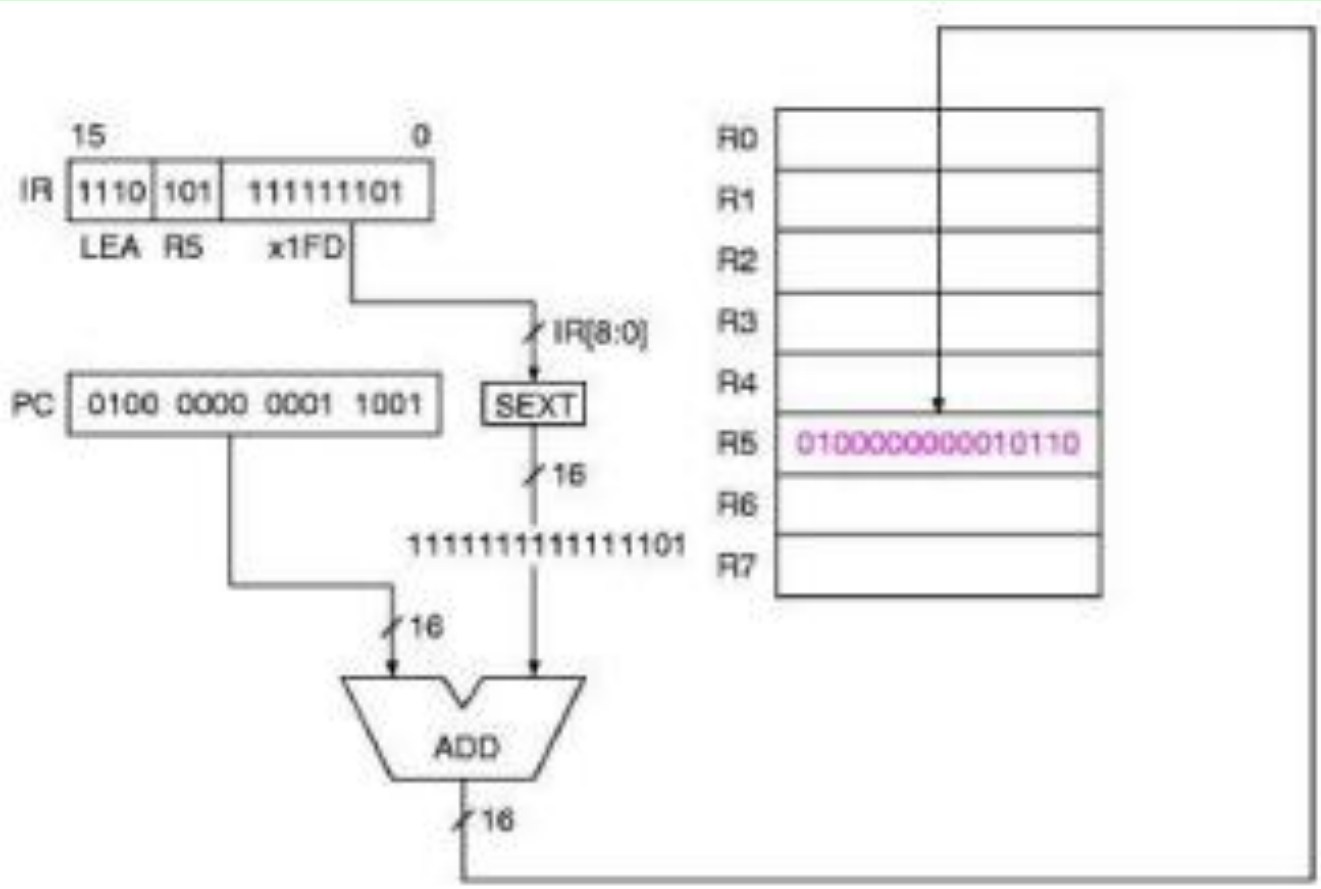
4.7.4 Immediate mode

Ví dụ 4.12: Nếu ô nhớ x4018 chứa lệnh LEA R5, #-3, và thanh ghi PC chứa x4018, ta có lệnh



CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu



Hình 4.9 Đường truyền dữ liệu khi thực hiện lệnh LEA R5, #-3

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.7 Nhóm lệnh di chuyển dữ liệu

4.7.5 Ví dụ: Giả sử các lệnh được chứa trong các ô nhớ từ địa chỉ x30F6 tới x30FC như hình 4.10 sau.

Địa chỉ lệnh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Thao tác
x30F6	1	1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	R1 ← PC – 3
x30F7	0	0	0	1	0	1	0	0	0	1	1	0	1	1	1	0	R2 ← R1 + 14
x30F8	0	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	M[x30F4] ← R2
x30F9	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	R2 ← 0
x30FA	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	R2 ← R2 + 5
x30FB	0	1	1	1	0	1	0	0	0	1	0	0	1	1	1	0	M[R1+14] ← R2
x30FC	1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	R3 ← M[M[x30F4]]

Hình 4.10 Ví dụ về các kiểu định vị

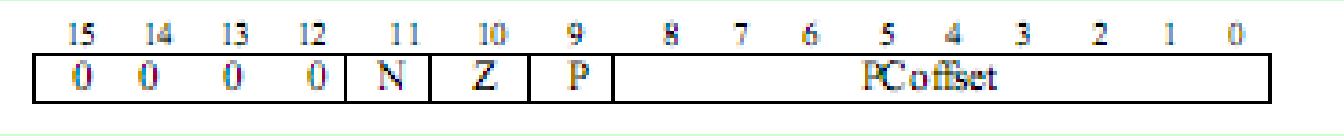
CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

Các lệnh điều khiển thay đổi trình tự các lệnh thực thi trong chương trình. LC-3 có năm mã lệnh thực hiện việc này: lệnh rẽ nhánh (Branch) có điều kiện, lệnh nhảy (Jump) không điều kiện, gọi chương trình con, TRAP, và lệnh trả về từ ngắt (Interrupt). Trong mục này chủ yếu chúng ta nói về lệnh rẽ nhánh có điều kiện, lệnh nhảy không điều kiện, và TRAP.

4.8.1 Lệnh rẽ nhánh có điều kiện

Định dạng của lệnh rẻ nhánh có điều kiện (opcode = 0000) như sau:



CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện

Trong LC-3, tất cả các lệnh mà có ghi trị vào các thanh ghi đa dụng sẽ đặt trị cho ba mã điều kiện (NZP) là ADD, AND, NOT, LD, LDI, LDR, và LEA.

Các mã điều kiện được lệnh rẽ nhánh có điều kiện sử dụng để xác định xem có thay đổi trình tự lệnh hay không; nghĩa là, xem sự thực thi lệnh có theo trình tự thường thấy như là kết quả của việc tăng thanh ghi PC trong phase FETCH của mỗi lệnh hay không.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện

Trong khi thực hiện phase EXECUTE, bộ xử lý khảo sát các mã điều kiện mà các bit tương ứng của nó trong lệnh là 1.

Nếu có một mã điều kiện mà được kiểm tra và thấy bằng 1, thanh ghi PC được nạp bằng địa chỉ có được trong phase EVALUATE ADDRESS.

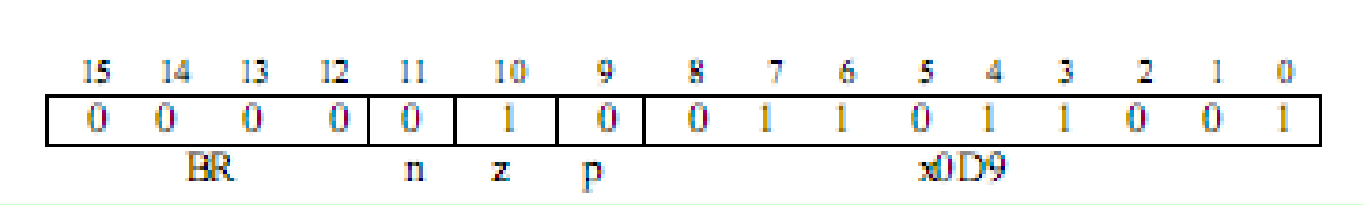
Ngược lại, khi không có một mã điều kiện mà được kiểm tra và thấy bằng 1, PC giữ trị không đổi.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện

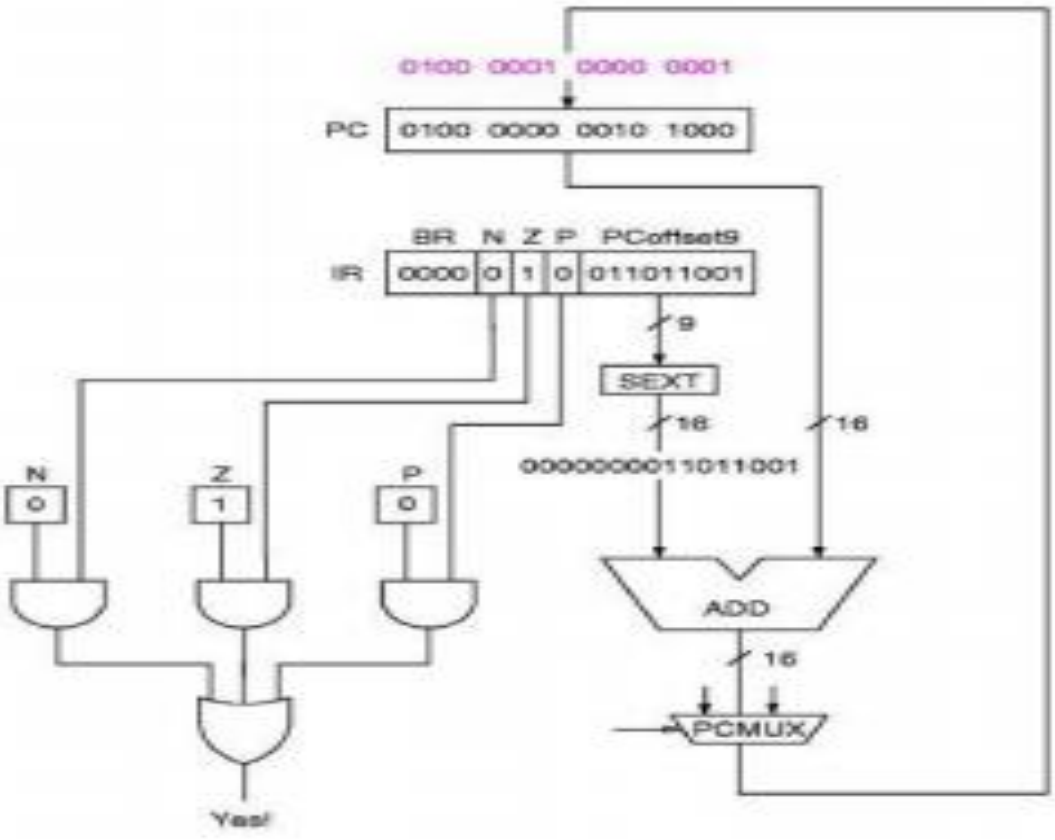
Ví dụ 4.13: Nếu giá trị cuối cùng được nạp vào một thanh ghi đa dụng nào đó là 0, thì lệnh hiện thời (ở ô nhớ x4027) dưới đây sẽ nạp thanh ghi PC bằng trị x4101 (x4028 + x0D9), tức lệnh kế cần thực thi sẽ ở ô nhớ x4101, chứ không phải ở ô nhớ x4028.



CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện



Hình 4.11 Đường truyền dữ liệu khi thực thi lệnh BRz x0D9

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện

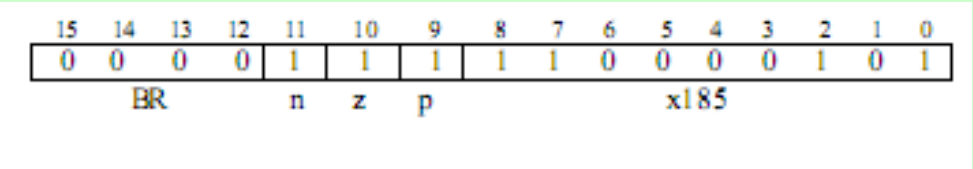
Nếu tất cả ba bit [11:9] đều là 1????

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.1 Lệnh rẽ nhánh có điều kiện

Ví dụ 4.14: Nếu lệnh sau đây



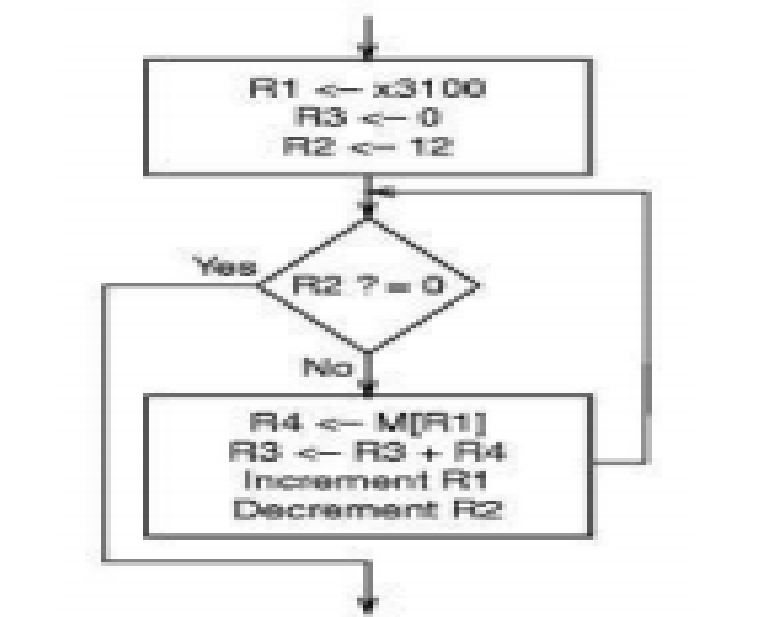
ở ô nhớ x507B được thực thi, thi thanh ghi PC được nạp trị x5201. Độc giả hãy suy nghĩ xem nếu tất cả ba bit [11:9] trong lệnh BR đều bằng không, điều gì sẽ xảy ra?

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.2 Ví dụ

Giả sử chúng ta đã có 12 ô nhớ từ x3100 tới x310B chứa 12 số nguyên mà chúng ta cần tính tổng.



4.12 Lưu đồ cộng 12 số nguyên

4.8 Nhóm lệnh điều khiển

4.8.2 Ví dụ

Địa chỉ lệnh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Thao tác
x3000	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	R1 ← 3100
x3001	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	R3 ← 0
x3002	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	R2 ← 0
x3003	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	0	R2 ← 12
x3004	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	BRz x300A
x3005	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	R4 ← M[R1]
x3006	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0	R3 ← R3 + R4
x3007	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	R1 ← R1 + 1
x3008	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1	R2 ← R2 – 1
x3009	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	BRnzp x3004

Hình 4.13 Chương trình hiện thực giải thuật ở hình 4.12

4.8 Nhóm lệnh điều khiển

4.8.3 Hai phương pháp điều khiển lặp

Chúng ta dùng thuật ngữ lặp (loop) để mô tả chuỗi lệnh cần được thực thi lặp đi lặp lại theo một cơ chế điều khiển nào đó.

Mỗi khi thân vòng lặp được thực hiện ta gọi là sự lặp lại (iteration) của vòng lặp.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.3 Hai phương pháp điều khiển lặp

Trong ví dụ ở mục trên, phương pháp biến đếm được sử dụng. Nếu chúng ta biết chính xác số lần lặp là n lần, thì đơn giản chúng ta chỉ cần gán cho biến đếm trị n, và sau mỗi lần lặp, chúng ta cần giảm biến đếm này và kiểm tra nó xem có là 0 hay chưa. Nếu chưa là 0, chúng ta đặt trị cho thanh ghi PC bằng địa chỉ của lệnh đầu thân vòng lặp, và tiếp tục sự lặp lại.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

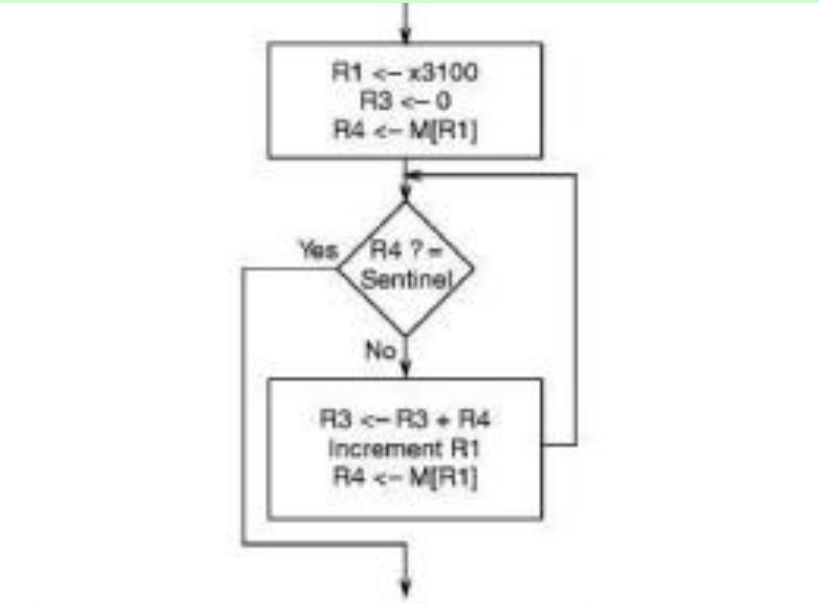
4.8.3 Hai phương pháp điều khiển lặp

Phương pháp thứ hai là dùng trị canh (sentinel). Cách này đặc biệt hiệu quả khi chúng ta không biết trước có bao nhiêu sự lặp lại cần phải được thực hiện. Thông thường quá trình lặp sẽ là dãy các sự kiện cần xử lý, chúng ta cần thêm vào dãy sự kiện này một sự kiện là kiểm tra một giá trị mà chúng ta biết trước hay theo quy ước là không bao giờ xuất hiện trong các sự kiện gốc. Nếu việc kiểm tra này xảy ra, tức việc lặp kết thúc. Giá trị đó được gọi là trị canh.

4.8 Nhóm lệnh điều khiển

4.8.4 Ví dụ

Giả sử chúng ta cần cộng các trị nguyên dương từ ô nhớ x3100 tới x310B. Khi đó chúng ta có thể dùng bất kỳ trị âm nào để làm trị canh. Giả sử trị canh được chứa ở ô nhớ x310C là -1.



Hình 4.14 Giải thuật sử dụng một trị canh điều khiển lặp

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.4 Ví dụ

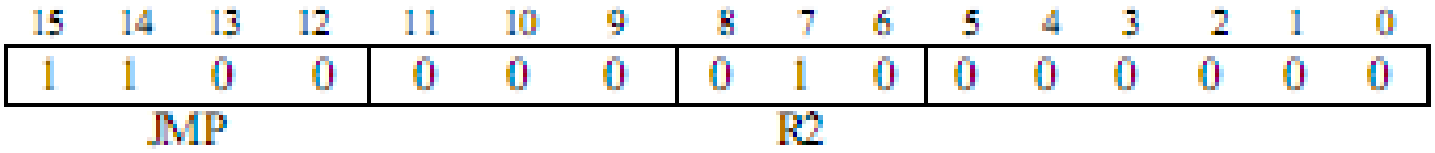
Địa chỉ lệnh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Thao tác
x3000	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	R1 ← x3100
x3001	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	R3 ← 0
x3002	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	R4 ← M[R1]
x3003	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	BRn x3008
x3004	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0	R3 ← R3 + R4
x3005	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	R1 ← R1 + 1
x3006	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	R4 ← M[R1]
x3007	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	BRnzp x3003

Hình 4.15 Chương trình hiện thực giải thuật ở hình 4.14

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.5 Lệnh JMP



Lệnh JMP nạp thanh ghi PC bằng trị của thanh ghi xác định bởi các bit [8:6] của lệnh. Nếu lệnh JMP nằm ở địa chỉ x4000, R2 chứa trị x6600, và PC chứa x4000, thì lệnh JMP ở x4000 sẽ được thực thi, làm cho lệnh kế tiếp là lệnh ở x6600. Vì các thanh ghi đều dài 16 bit, có thể mã hóa địa chỉ cho cả bộ nhớ 64K của ISA LC-3, nên lệnh JMP không có giới hạn cho lệnh kế cần được thực thi.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.8 Nhóm lệnh điều khiển

4.8.6 Lệnh TRAP

Lệnh TRAP (opcode = 1111) cho phép lấy dữ liệu vào và xuất dữ liệu ra khỏi máy tính và có định dạng như sau.

Lệnh này thay đổi trị của thanh ghi PC, làm nó chỉ tới một vị trí trong bộ nhớ mà là một phần của hệ điều hành để hệ điều hành thực hiện một tác vụ nào đó nhân danh chương trình đang được thực thi. Theo kiểu nói của ngôn ngữ hệ điều hành, chúng ta nói lệnh TRAP gọi một dịch vụ của hệ điều hành (service call). Các bit [7:0] của lệnh TRAP tạo nên một trapvector 8 bit, xác định dịch vụ cần gọi mà chương trình muốn hệ điều hành thực hiện.

4.8 Nhóm lệnh điều khiển

4.8.6 Lệnh TRAP

Một số dịch vụ chúng ta cần biết để sử dụng ngay là :

- Nhập một ký tự từ bàn phím: trapvector = x23
- Xuất một ký tự ra màn hình: trapvector = x21
- Kết thúc chương trình: trapvector = x25.

Khi hệ điều hành hoàn tất dịch vụ, bộ đếm chương trình PC sẽ được đặt trở lại địa chỉ của lệnh ngay sau lệnh TRAP trong chương trình đang được thực thi, và chương trình tiếp tục chạy. Đây cũng là khả năng thực hiện giao tiếp của LC-3 giữa hệ điều hành và chương trình đang được thực thi.

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.9 Ba cấu trúc lệnh trong LC-3

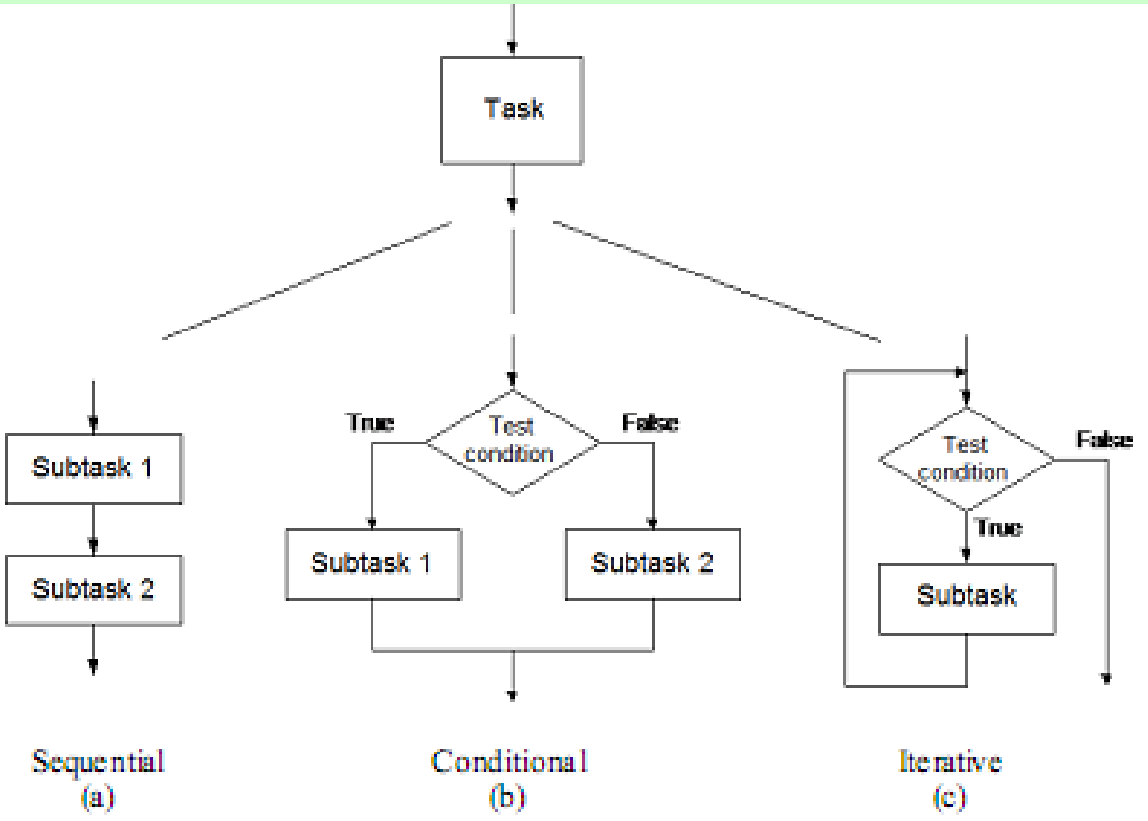
4.9.1 Ba cấu trúc cơ bản trong lập trình có cấu trúc

Trong lập trình có ba cấu trúc lệnh cơ bản để thực hiện các tác vụ cần thiết, đó là các cấu trúc tuần tự (sequential), điều kiện (conditional), và lặp (iterative). Hình 4.16 trình bày ba cấu trúc cơ bản triển khai tác vụ cần thực thi (task).

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.9 Ba cấu trúc lệnh trong LC-3

4.9.1 Ba cấu trúc cơ bản trong lập trình có cấu trúc



Hình 4.16 Ba cấu trúc lệnh của lập trình cấu trúc

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.9 Ba cấu trúc lệnh trong LC-3

4.9.2 Ba cấu trúc trong LC-3

Hình 4.17 Ba cấu trúc lệnh trong LC-3

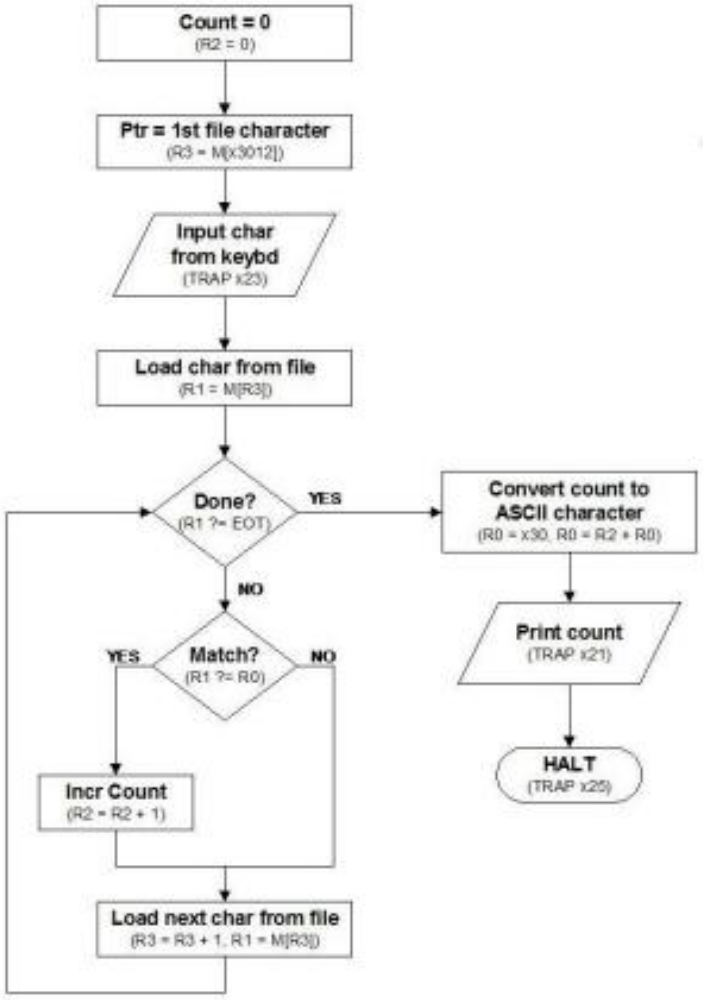
CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.10 Một ví dụ

Đếm số lần xuất hiện của một ký tự xác định trước (được nhập từ bàn phím) trong một mảng ký tự (file) cho trước. Sau đó hiển thị số lần xuất hiện này ra màn hình(chấp nhận số lần xuất hiện tối đa của một ký tự là 9)

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.10 Một ví dụ



Hình 4.18 Giải thuật đếm số lần xuất hiện của ký tự

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.10 Một ví dụ

Địa chỉ lệnh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Thao tác
x3000	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	R2 ← 0
x3001	0	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	R3 ← M[x3012]
x3002	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	1	TRAP x23
x3003	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	R1 ← M[R3]
x3004	0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0	R4 ← R1 – 4
x3005	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	BRz x300E
x3006	1	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	R1 ← NOT R1
x3007	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	R1 ← R1 + 1
x3008	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	R1 ← R1 + R0
x3009	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	BRnp x300B
x300A	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	R2 ← R2 + 1
x300B	0	0	0	1	0	1	1	0	1	1	1	0	0	0	0	1	R3 ← R3 + 1
x300C	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	R1 ← M[R3]
x300D	0	0	0	0	1	1	1	1	1	1	1	1	0	1	1	0	BRnzp x3004

CHƯƠNG 4
MÔ HÌNH VON NEUMANN
VÀ KIẾN TRÚC TẬP LỆNH LC-3

4.10 Một ví dụ

x300E	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	R0 ← M[x3013]
x300F	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	R0 ← R0 + R2
x3010	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1	TRAP x21
x3011	1	1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	TRAP x25
x3012	Địa chỉ bắt đầu của file																
x3013	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	Bảng ASCII

Hình 4.19 Chương trình ngôn ngữ máy hiện thực giải thuật hình 4.16