TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN

KIẾN TRÚC MÁY TÍNH

Giảng viên: TS. Đoàn Thị Quế

Bộ môn: Mạng và an toàn thông tin

Chương 9. Tập lệnh: Đặc điểm và chức năng

9.1. Đặc điểm tập lệnh

9.2. Các kiểu toán hạng

9.3. Các kiểu dữ liệu Intel x86 và ARM

9.4. Các loại hoạt động

9.5. Các loại hoạt động Intel x86 và ARM

9.1 Đặc điểm tập lệnh

a. Thành phần của lệnh máy

b. Biểu diễn lệnh

c. Các loại lệnh

d. Số lượng địa chỉ

e. Thiết kế tập lệnh

9.1 Đặc điểm tập lệnh

- Hoạt động (operation) của bộ xử lý được quyết định bởi các lệnh nó thực hiện, đó là các lệnh máy tính (*machine instructions* hay *computer instructions)*
  - Ví dụ: Lệnh STORE: lưu trữ dữ liệu vào bộ nhớ
- Tập hợp các lệnh khác nhau mà bộ xử lý có thể thực hiện được gọi là **tập lệnh (*instruction set)*** của bộ xử lý
- Mỗi lệnh phải có những thông tin cần thiết cho bộ xử lý thực hiện hoạt động:
  - Chức năng của lệnh
  - Toán hạng của lệnh
  - Nơi cất kết quả
  - Địa chỉ của lệnh tiếp theo

a. Các thành phần của lệnh máy

- **Mã lệnh** - Operation code (opcode): chỉ ra hoạt động (operation: hoạt động/phép toán) được thực hiện thông qua một mã nhị phân được gọi là mã lệnh (opcode)
- **Tham chiếu toán hạng nguồn:** Mỗi hoạt động có thể tham chiếu đến một hoặc nhiều toán hạng để lấy dữ liệu đầu vào cho hoạt động: các toán hạng này được gọi là **toán hạng nguồn**
- **Tham chiếu toán hạng kết quả (toán hạng đích):** Hoạt động có thể đưa ra một kết quả
- **Tham chiếu lệnh tiếp theo:** Một số hoạt động có thể rẽ nhánh đến một câu lệnh ở vị trí khác → nói cho bộ xử lý nơi để lấy lệnh tiếp theo sau khi việc thực thi lệnh hiện tại hoàn thành
- VD: lệnh của máy IAS (20b: 8b opcode và 12b địa chỉ toán hạng)

Lệnh máy	Mô tả hợp ngữ	Công việc
00100001 0001 1010 0000 Mã lệnh 1A0	STOR M(1A0)	Truyền dữ liệu từ AC vào ngăn nhớ có địa chỉ 1A0 trong bộ nhớ. AC: toán hạng nguồn (ngầm định) 1A0: toán hạng đích

Các toán hạng nguồn và kết quả có thể ở một trong bốn vùng sau:

1) Bộ nhớ chính hoặc bộ nhớ ảo

- Toán hạng nguồn/kết quả có thể là một vị trí bộ nhớ chính hoặc bộ nhớ ảo. Trong lệnh phải chứa địa chỉ bộ nhớ chính hoặc bộ nhớ ảo của toán hạng đó

2) Thanh ghi

- Toán hạng nguồn hoặc kết quả có thể là các thanh ghi. Một VXL chứa một hoặc nhiều thanh ghi, mỗi thanh ghi được gán cho một tên hoặc số riêng.
- Một lệnh có thể tham chiếu đến các thanh ghi này.

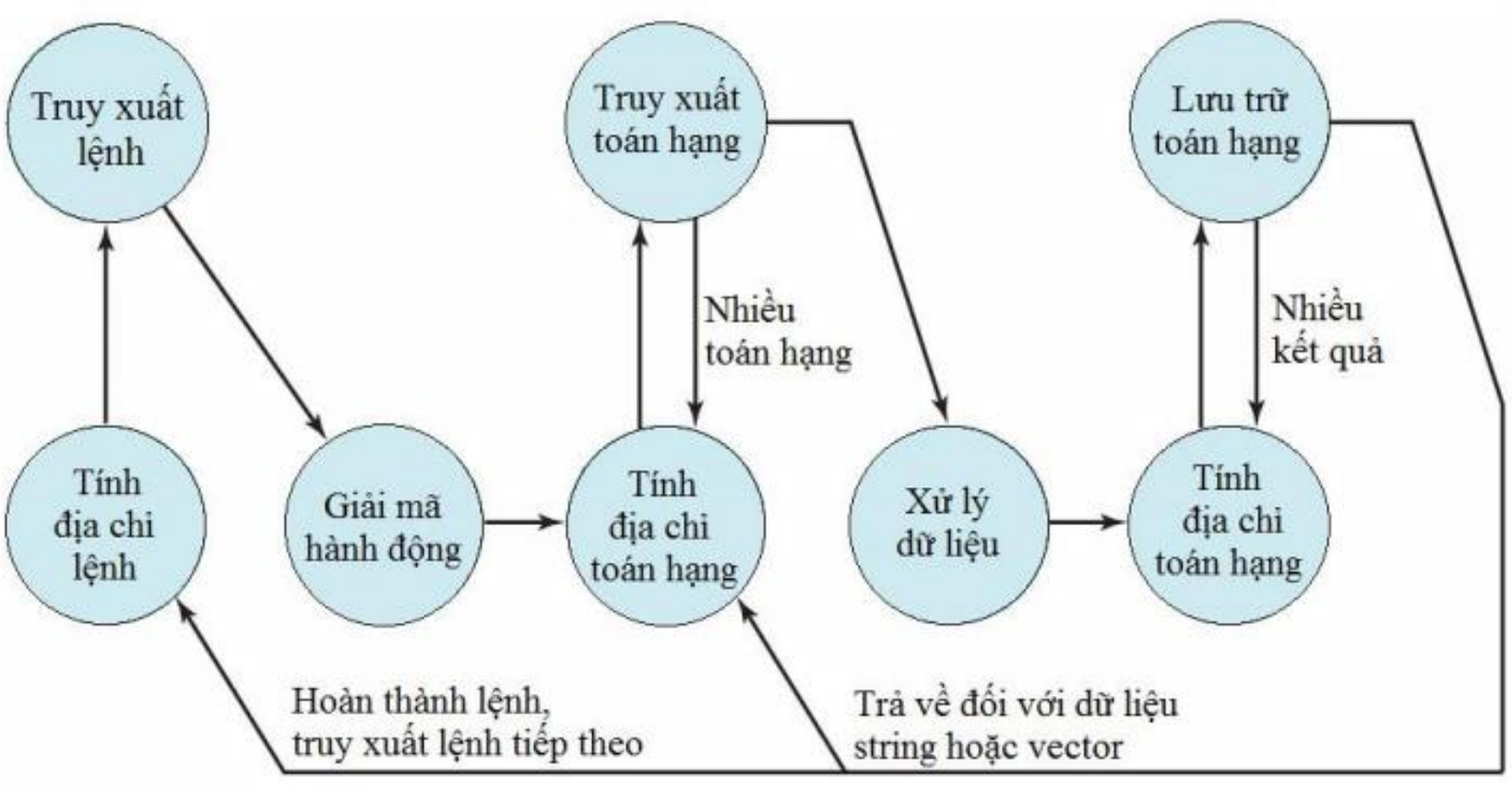
3) Tức thì

- Giá trị của toán hạng có thể được đưa trực tiếp vào trong câu lệnh

4) Thiết bị vào/ra (I/O)

- Dữ liệu có thể lấy từ (hoặc ghi ra) một thiết bị I/O → lệnh phải chỉ ra địa chỉ thiết bị và module vào/ra tương ứng

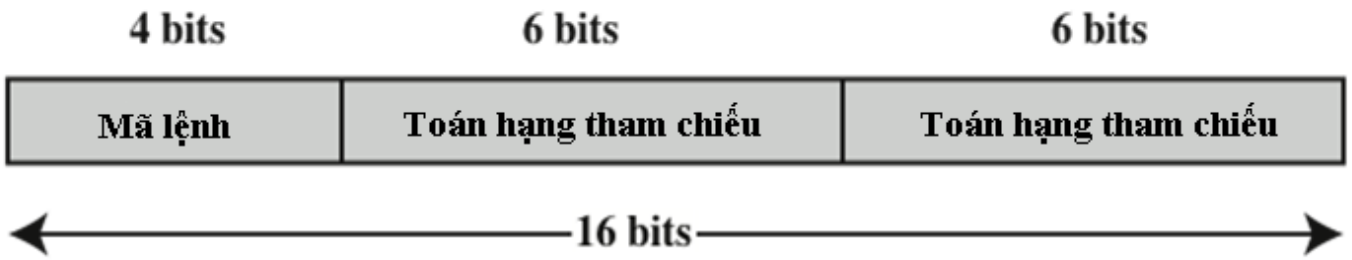
Biểu đồ chu kỳ lệnh



Hình 10.1 Sơ đồ trạng thái chu kỳ lệnh

b. Biểu diễn lệnh

- Trong máy tính, mỗi câu lệnh được biểu diễn bằng một chuỗi bit nhị phân
- Câu lệnh được chia ra thành các trường tương ứng với các thành phần cấu thành của lệnh
- Ví dụ



- Trong các tài liệu, để dễ hiểu, lệnh thường được biểu diễn dưới dạng các ký hiệu thay vì các bit nhị phân.

b. Biểu diễn lệnh

- Mã lệnh được biểu diễn bằng các từ viết tắt, gợi nhớ về hành động tương ứng.
  - Ví dụ:
    - ADD Cộng
    - SUB Trừ
    - MUL Nhân
    - DIV Chia
    - LOAD Nạp dữ liệu từ bộ nhớ
    - STOR Lưu trữ dữ liệu vào bộ nhớ
- Các toán tử cũng được biểu diễn bằng ký tự.
  - Ví dụ, lệnh
    - ADD R, Y

Ví dụ:

Ngôn ngữ lập trình bậc cao được đưa ra để giúp công việc của lập trình viên thuận lợi hơn

Ví dụ câu lệnh: X=X+Y viết bằng C++ nếu dịch sang tập lệnh IAS sẽ gồm các lệnh như sau

0010 0000 0001 0000	LOAD X	Đọc X từ bộ nhớ vào thanh ghi AC
0101 0000 0001 0001	ADD Y	Đọc Y từ bộ nhớ, cộng Y với AC, kết quả ghi vào AC
1000 0000 0001 0000	STOR X	Ghi AC vào X trong bộ nhớ

Trong đó: X, Y là biến, có địa chỉ BN: X: 0000 0001 0000
Y: 0000 0001 0001

c. Các loại lệnh: chia thành 4 nhóm

- Xử lý dữ liệu: các lệnh số học và logic
  - Các lệnh số học cung cấp khả năng tính toán để xử lý dữ liệu số
  - Các lệnh logic (Boolean) hoạt động trên các bit, cung cấp khả năng xử lý bất kỳ loại dữ liệu nào. Các lệnh này chủ yếu thực thi với các bit trên thanh ghi
- Lưu trữ dữ liệu
  - Các lệnh đọc/ghi dữ liệu từ/vào thanh ghi hoặc bộ nhớ
- Di chuyển dữ liệu:
  - Gồm các lệnh vào/ra: được sử dụng để truyền chương trình và dữ liệu vào bộ nhớ và các kết quả tính toán được trở lại cho người dùng

  Ví dụ: chương trình user và dữ liệu (lưu trữ ở ổ cứng) được nạp vào RAM
- Điều khiển: gồm các lệnh kiểm tra và rẽ nhánh
  - Các lệnh kiểm tra được sử dụng để kiểm tra giá trị của dữ liệu hoặc trạng thái của một phép toán
  - Các lệnh rẽ nhánh được dùng để rẽ nhánh tập lệnh khác nhau tùy thuộc vào **điều kiện** cụ thể

d. Số lượng các địa chỉ

- Một thuộc tính quan trọng của tập lệnh là **số lượng địa chỉ**
- Tùy thuộc vào các lệnh khác nhau sẽ có số lượng toán hạng khác nhau
  - Như phần trên đã đề cập, các toán hạng có thể là các vị trí nhớ (trong bộ nhớ chính) hoặc I/O (I/O port), được đặc trưng bởi một địa chỉ logic
- Vậy, số lượng địa chỉ tối đa trong một lệnh là bao nhiêu:
  - Các lệnh số học và logic: cần tối đa 4 địa chỉ: 2 địa chỉ toán hạng nguồn, 1 địa chỉ toán hạng đích, 1 địa chỉ toán hạng truy xuất câu lệnh tiếp theo.
  - Số lượng địa chỉ càng nhiều thì kích thước lệnh càng lớn
  - Với hầu hết các hệ vi xử lý, số lượng địa chỉ là 1, 2 hoặc 3. Lệnh tiếp theo được ngầm định truy xuất thông qua thanh ghi PC (program counter register)

d. Số lượng các địa chỉ (tiếp)

Ví dụ các lệnh tính toán biểu thức $Y=(A-B)/(C+(D\times E))$ trong 3 trường hợp:

Lệnh		Giải thích
SUB	Y, A, B	Y ← A – B
MPY	T, D, E	T ← D × E
ADD	T, T, C	T ← T + C
DIV	Y, Y, T	Y ← Y ÷ T

(a) Lệnh có ba địa chỉ

Lệnh		Giải thích
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y – B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

(b) Lệnh có hai địa chỉ

Lệnh		Giải thích
LOAD	D	AC ← D
MPY	E	AC ← AC × E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC – B
DIV	Y	AC ← AC ÷ Y
STOR	Y	Y ← AC

(c) Lệnh có một địa chỉ

Hình 10.3 Chương trình tính biểu thức $Y = \frac{A - B}{C + (D \times E)}$

- Trường hợp lệnh 3 địa chỉ:
  - Kích thước lệnh dài
  - Toán hạng 1, toán hạng 2, kết quả
  - T: vị trí bộ nhớ tạm thời để lưu trữ kết quả
- Trường hợp lệnh 1 địa chỉ:
  - Một toán hạng ngầm định là thanh ghi AC
  - Phổ biến ở các hệ VXL đơn giản, đời đầu

- Số lượng các địa chỉ càng ít thì số lượng các câu lệnh để tính toán biểu thức càng nhiều

- *Lưu ý: Trường hợp lệnh 0 địa chỉ được áp dụng cho một tổ chức bộ nhớ đặc biệt được gọi là ngăn xếp.*

Bảng 10.1 Cách sử dụng địa chỉ lệnh (Lệnh không rẽ nhánh)

Số lượng địa chỉ	Biểu diễn ký tự	Giải thích
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
0	OP	T ← (T - 1) OP T

AC = Thanh ghi tích lũy

T = đỉnh ngăn xếp

(T – 1) = phần tử thứ hai trong ngăn xếp

A, B, C = các vị trí bộ nhớ hoặc thanh ghi

Nhận xét: Số lượng các địa chỉ

- Số lượng địa chỉ trong mỗi lệnh là một yếu tố cơ bản đối với thiết kế VXL.
- Lệnh càng ít địa chỉ → kích thước lệnh ngắn hơn →VXL ít phức tạp hơn → chương trình cần nhiều lệnh hơn để thực hiện một công việc → mất thời gian hơn
- Lệnh một địa chỉ, lập trình viên thường chỉ có sẵn một thanh ghi đa năng: *thanh ghi AC.*
- Với các hệ VXL cho phép lệnh nhiều địa chỉ thường có nhiều thanh ghi đa năng. Điều này cho phép một số hoạt động được thực hiện chỉ trong các thanh ghi → không cần truy xuất BNC → tốc độ nhanh hơn.

→ Hầu hết các hệ VXL hiện đại sử dụng kết hợp các cấu trúc lệnh hai và ba địa chỉ.

e. Thiết kế tập lệnh

- Tập lệnh định nghĩa các chức năng được thực hiện bởi VXL
- Là phương tiện của người lập trình trong việc điều khiển VXL
- Các vấn đề thiết kế cơ bản:
  - **Danh sách các hoạt động**: bao nhiêu hoạt động và hoạt động nào được đưa ra? Độ phức tạp của các hoạt động như thế nào?
  - **Các kiểu dữ liệu**: các kiểu dữ liệu mà các hoạt động tham chiếu đến
  - **Cấu trúc lệnh**: độ dài lệnh theo bit, số lượng địa chỉ, kích thước của các trường khác nhau, v.v ...
  - **Các thanh ghi**: số lượng các thanh ghi của VXL có thể được tham chiếu đến bởi lệnh và chức năng của chúng
  - **Chế độ địa chỉ**: các cách để định ra địa chỉ của toán hạng

Chương 9. Tập lệnh: Đặc điểm và chức năng

9.1. Đặc điểm tập lệnh

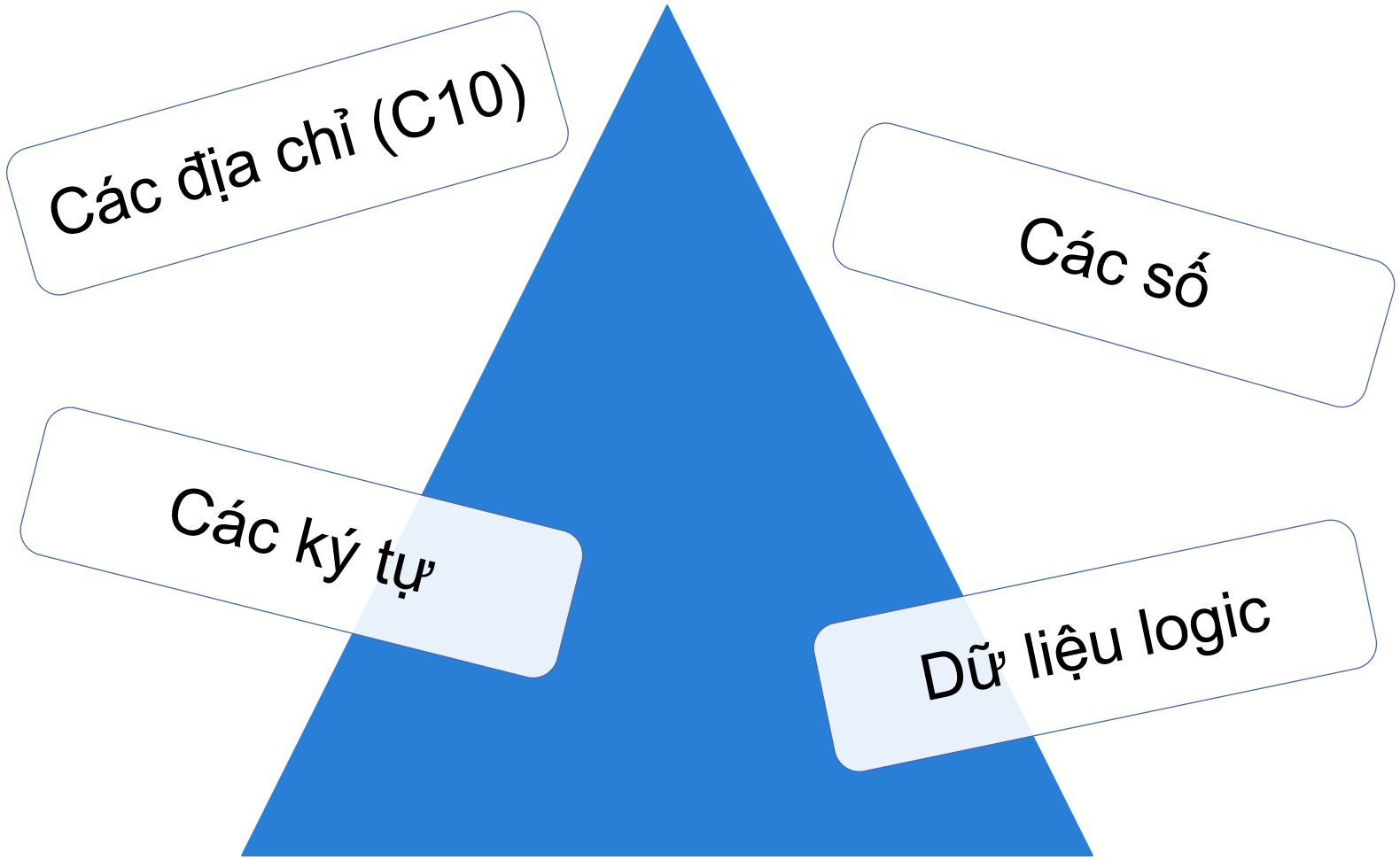
9.2. Các kiểu toán hạng

9.3. Các kiểu dữ liệu Intel x86 và ARM

9.4. Các loại hoạt động

9.5. Các loại hoạt động Intel x86 và ARM

9.2 Các kiểu toán hạng



Dữ liệu kiểu số

- Tất cả các ngôn ngữ máy đều có dữ liệu dạng số
- Các số được lưu trữ trong máy tính đều hữu hạn:
  - Hữu hạn về độ lớn của các số biểu diễn trên máy
  - Hữu hạn về độ chính xác đối với số dấu phẩy động

- Ba kiểu dữ liệu số thông thường trong máy tính:
  1. Số nguyên nhị phân hoặc số nhị phân dấu chấm tĩnh
  2. Số nhị phân dấu chấm động
  3. Số thập phân đóng
     - Mỗi chữ số thập phân được biểu diễn bởi một mã 4 bit
       - 0 = 0000, 1 = 0001,…, 8 = 1000 và 9 = 1001, mã này được gọi là Binary Coded Decimal (BCD)
     - Chiều dài mã thường là bội của 8b
       - Ví dụ: Mã thập phân đóng của 246 là 0000 0010 0100 0110
     - Số thập phân đóng có dấu: Dấu dương (+): 1100, dấu âm (-): 1101

Dữ liệu kiểu ký tự

- Một trong những dạng dữ liệu cơ bản là văn bản (text) hoặc xâu ký tự (character strings)
- Dữ liệu văn bản dưới dạng ký tự không thể lưu trữ hoặc truyền qua hệ thống xử lý dữ liệu và truyền thông vì các hệ thống này được thiết kế cho dữ liệu nhị phân → sử dụng bảng mã
- Bảng mã mã hóa ký tự được sử dụng phổ biến nhất là bảng mã IRA (International Reference Alphabet)
  - Còn được gọi ở Mỹ là bảng mã ASCII (American Standard Code for Information Interchange)
- Một bảng mã khác được sử dụng để mã hoá các ký tự bảng mã EBCDIC (Extended Binary Coded Decimal Interchange Code) được sử dụng trong các máy mainframe của IBM

Dữ liệu logic

- Một khối n-bit gồm n phần tử dữ liệu 1 bit, mỗi phần tử có giá trị 0 hoặc 1
- Việc xét dữ liệu theo hướng bit có hai ưu điểm:
  - Đôi khi, ta muốn lưu trữ một mảng các bit nhị phân hoặc dữ liệu Boolean/nhị phân, trong đó mỗi phần tử chỉ nhận giá trị 1 (đúng) hoặc 0 (sai). Với kiểu dữ liệu logic, bộ nhớ lưu trữ điều này hiệu quả nhất
  - Trong một số trường hợp chúng ta cần thao tác với các bit
    - Trường hợp phép toán dấu chấm động: dịch các bit có nghĩa
    - Trường hợp chuyển đổi từ mã IRA thành mã thập phân đóng gói: trích xuất 4 bit bên phải của mỗi byte

Chương 9. Tập lệnh: Đặc điểm và chức năng

9.1. Đặc điểm tập lệnh

9.2. Các kiểu toán hạng

9.3. Các kiểu dữ liệu Intel x86 và ARM

9.4. Các loại hoạt động

9.5. Các loại hoạt động Intel x86 và ARM

9.3 Các kiểu dữ liệu Intel x86 và ARM

a. Intel x86

Kiểu dữ liệu	Mô tả
General	Các vị trí bộ nhớ kích thước byte, word (16 bits), doubleword (32 bits), quadword (64 bits), và double quadword (128 bits) với dữ liệu nhị phân bất kỳ
Integer	Giá trị nhị phân có dấu lưu trữ trong một byte, word, hoặc doubleword, sử dụng dạng biểu diễn bù 2
Ordinal	Một số nguyên không dấu lưu trữ trong một byte, word, hoặc doubleword.
Unpacked binary coded decimal (BCD)	Biểu diễn một ký tự BCD trong khoảng từ 0 đến 9, với mỗi ký tự dùng một byte
Packed BCD	Biểu diễn 2 ký tự BCD trong 1 byte, một packed BCD có dải giá trị từ 0 đến 99
Near pointer	Địa chỉ hiệu dụng 16-bit, 32-bit, hoặc 64-bit biểu diễn độ lệch (offset) trong một phân đoạn. Được sử dụng cho tất cả các con trỏ trong bộ nhớ không phân đoạn và cho các tham chiếu trong một đoạn của bộ nhớ phân đoạn
Far pointer	Địa chỉ logic gồm 16-bit trỏ tới một đoạn (segment) và một địa chỉ lệch offset 16, 32, hoặc 64 bits. Far pointers được sử dụng để tham chiếu bộ nhớ trong mô hình bộ nhớ phân đoạn
Bit field	Một chuỗi bit liên tục trong đó mỗi bit được coi như một đơn vị độc lập. Chuỗi bit có thể bắt đầu tại bất cứ vị trí nào trong bất cứ byte nào và có thể chứa tới 32 bit
Bit string	Một dãy bit liên tục, gồm từ 0 đến 232 - 1 bit.
Byte string	Một dãy byte, word hoặc doublewords liên tục gồm từ 0 đến 232 - 1 byte.
Floating point	Xem hình 10.4.
Packed SIMD (single instruction, multiple data)	Các kiểu dữ liệu Packed 64-bit and 128-bit

Định dạng dữ liệu số x86

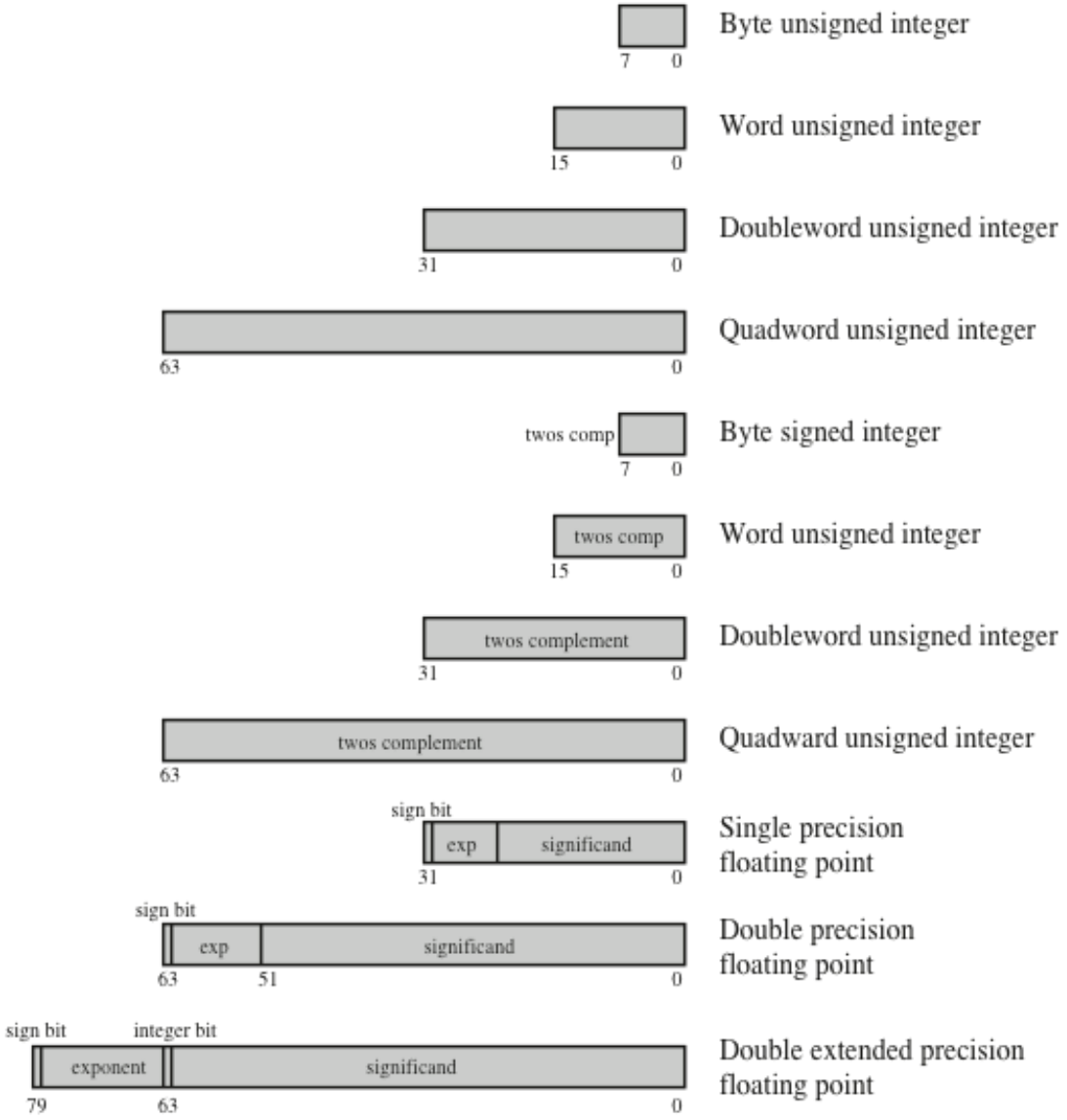


Figure 12.4 x86 Numeric Data Formats

Các kiểu dữ liệu SIMD (Single-Instruction-Multiple-Data)

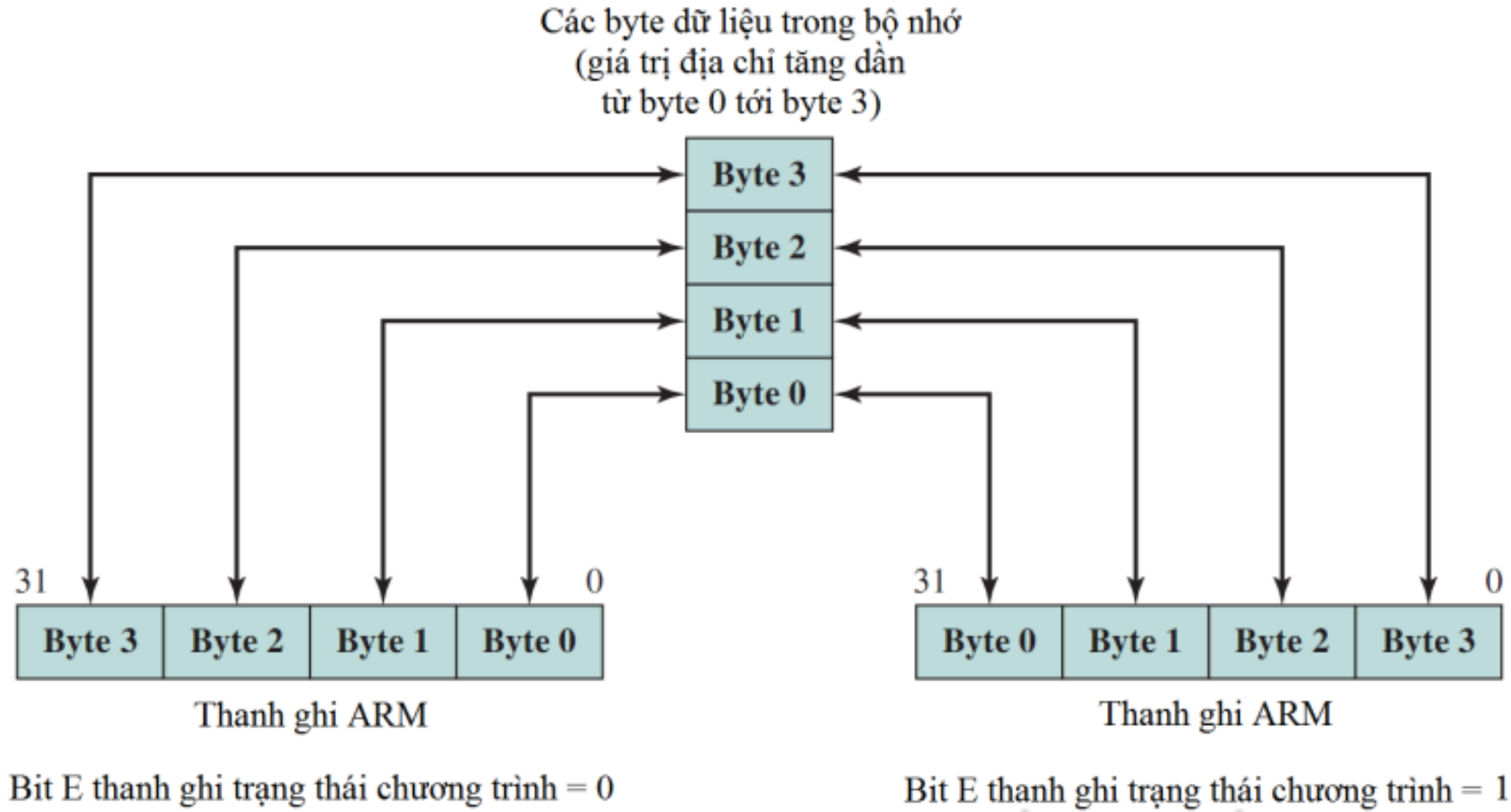
- Dùng cho kiến trúc x86 như là 1 phần của mở rộng tập lệnh để tối ưu hóa hiệu năng của các ứng dụng đa phương tiện
- Các mở rộng tập lệnh gồm MMX (multimedia extensions) và SSE (streaming SIMD extensions)
- Các kiểu dữ liệu SMID:
  - Byte đóng gói và số nguyên byte đóng gói
  - Word đóng gói và số nguyên word đóng gói
  - Doubleword đóng gói và số nguyên doubleword đóng gói
  - Quadword đóng gói và số nguyên quadword đóng gói
  - Packed single-precision floating-point and packed doubleprecision floating-point

b. Các kiểu dữ liệu của ARM

- Vi xử lý ARM hỗ trợ các kiểu dữ liệu có kích thước:
  - 8b (byte)
  - 16b (halfword)
  - 32b (word)
- Đối với tất cả ba kiểu dữ liệu đều hỗ trợ việc biểu diễn số nguyên không dấu (số nguyên dương)
- Tất cả ba kiểu dữ liệu cũng có thể được sử dụng cho số nguyên biểu diễn bù 2
- ARM không hỗ trợ phần cứng cho biểu diễn dấu chấm động. Các phép toán cho dấu chấm động phải được thực hiện bằng phần mềm

Hỗ trợ chuyển đổi Endian trong ARM

- Khái niệm Endian: cách tổ chức dữ liệu trong bộ nhớ máy tính
- Có 2 loại endian:
  - Little Endian: byte có giá trị nhỏ nhất lưu trữ ở vị trí nhớ có địa chỉ nhỏ nhất
  - Big Endian: byte có giá trị lớn nhất lưu trữ ở vị trí nhớ có địa chỉ nhỏ nhất
- ARM cho phép chuyển đổi giữa hai dạng endian: sử dụng E-bit trong thanh ghi PS (thanh ghi trạng thái chương trình)
  - E-bit = 1: Big endian
  - E-bit = 0: Little endian



Hình 10.5 Hỗ trợ Endian trong ARM – Tải/lưu trữ word với bit E

Chương 9. Tập lệnh: Đặc điểm và chức năng

9.1. Đặc điểm tập lệnh

9.2. Các kiểu toán hạng

9.3. Các kiểu dữ liệu Intel x86 và ARM

9.4. Các loại hoạt động

9.5. Các loại hoạt động Intel x86 và ARM

9.4 Các loại hoạt động (lệnh)

- Có rất nhiều các loại lệnh khác nhau đối với mỗi thế hệ máy tính. Tuy nhiên, một số loại chung đối với tất cả các máy tính như sau:
  a. Các lệnh truyền dữ liệu
  b. Các lệnh tính toán số học
  c. Các lệnh logic
  d. Các lệnh chuyển đổi
  e. Các lệnh vào/ra
  f. Các lệnh điều khiển hệ thống
  g. Các lệnh truyền điều khiển

a. Truyền dữ liệu

- Đây là lệnh cơ bản nhất của tất cả các hệ máy tính
- Cần phải định rõ:
  - Địa chỉ của các toán hạng nguồn và đích, loại vị trí: bộ nhớ, thanh ghi, ngăn xếp
  - Kích thước của dữ liệu được truyền
  - Chế độ định địa chỉ đối với mỗi toán hạng
- Các lệnh cơ bản

Truyền dữ liệu	MOVE	Copy dữ liệu từ nguồn đến đích
	LOAD	Nạp dữ liệu từ bộ nhớ đến bộ xử lý
	STORE	Cất dữ liệu từ bộ xử lý đến bộ nhớ
	EXCHANGE	Trao đổi nội dung của nguồn và đích
	CLEAR	Chuyển các bit 0 vào toán hạng đích
	SET	Chuyển các bit 1 vào toán hạng đích
	PUSH	Cất nội dung toán hạng nguồn vào ngăn xếp
	POP	Lấy nội dung đỉnh ngăn xếp đưa đến toán hạng đích

b. Số học

- Hầu hết các máy đều cung cấp các phép toán số học cơ bản: cộng, trừ, nhân, chia.
  - Các phép toán này luôn được cung cấp cho các số nguyên có dấu (dấu chấm tĩnh).
  - Một số máy có các phép toán trên với:
    - Số thực dấu chấm động
    - Số thập phân đóng
- Một số phép toán khác có dạng lệnh một toán hạng:
  - Tuyệt đối: Tính giá trị tuyệt đối của một toán hạng
  - Phép đảo: Đổi dấu một toán hạng
  - Phép tăng: Cộng toán hạng thêm 1 đơn vị
  - Phép giảm: Trừ toán hạng đi 1 đơn vị

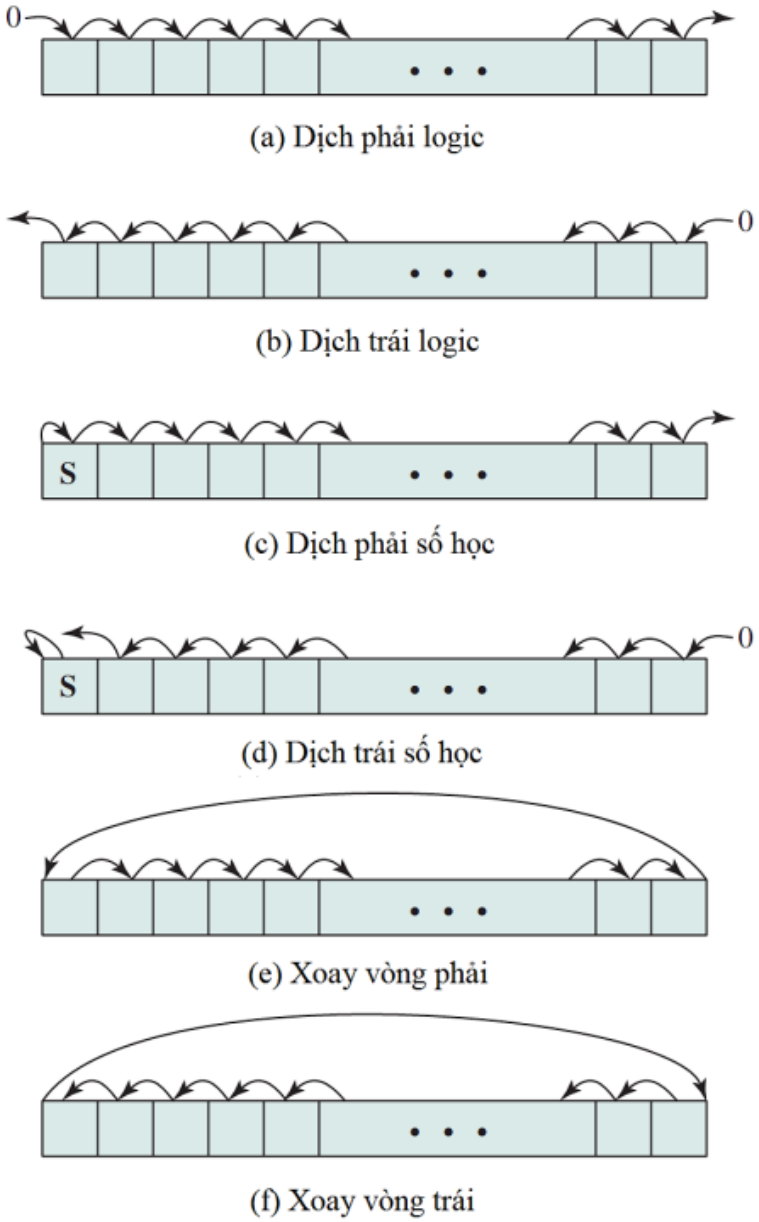
Một số lệnh số học

Xử lý số học	ADD	Cộng hai toán hạng
	SUBTRACT	Trừ hai toán hạng
	MULTIPLY	Nhân hai toán hạng
	DIVIDE	Chia hai toán hạng
	ABSOLUTE	Lấy trị tuyệt đối toán hạng
	NEGATE	Đổi dấu toán hạng (lấy bù 2)
	INCREMENT	Tăng toán hạng thêm 1
	DECREMENT	Giảm toán hạng đi 1

c. Phép toán logic

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

Xử lý logic	AND	Thực hiện phép AND hai toán hạng
	OR	Thực hiện phép OR hai toán hạng
	NOT	Đảo bit của toán hạng (lấy bù 1)
	XOR	Thực hiện phép XOR hai toán hạng
	TEST	Kiểm tra điều kiện cụ thể; thiết lập cờ dựa trên kết quả
	COMPARE	So sánh logic hoặc số học của hai hoặc nhiều toán hạng; thiết lập cờ dựa trên kết quả
	SHIFT	Dịch trái (phải) toán hạng
	ROTATE	Quay vòng trái (phải) toán hạng



Hình 10.6 Phép dịch và xoay vòng

Bảng 10.6 Ví dụ cho phép dịch và xoay vòng

Đầu vào	Hành động	Kết quả
10100110	Dịch phải logic (3 bit)	00010100
10100110	Dịch trái logic (3 bit)	00110000
10100110	Dịch phải số học (3 bit)	11110100
10100110	Dịch trái số học (3 bit)	10110000
10100110	Xoay vòng phải (3 bit)	11010100
10100110	Xoay vòng trái (3 bit)	00110101

d. Chuyển đổi

- Lệnh chuyển đổi: thay đổi định dạng hoặc tác động vào định dạng của dữ liệu
  - Ví dụ: chuyển đổi từ thập phân sang nhị phân

e. Vào/ra

- Các lệnh vào/ra liên quan đến:
  - Cơ chế địa chỉ:
    - I/O chương trình, ánh xạ riêng biệt - Isolated programmed I/O
    - I/O chương trình, ánh xạ bộ nhớ - Memory-mapped programmed I/O
  - Cơ chế DMA
  - Cơ chế điều khiển I/O sử dụng bộ xử lý vào ra
- Nhiều tập lệnh của các hệ VXL chỉ cung cấp một vài lệnh I/O với các hoạt động cụ thể được xác định bởi các tham số, các mã hoặc các từ lệnh

Điều khiển vào/ra	INPUT	Truyền dữ liệu từ một cổng hoặc thiết bị I/O xác định đến đích (VD: Bộ nhớ chính hoặc thanh ghi bộ xử lý)
	OUTPUT	Truyền dữ liệu từ nguồn xác định đến cổng hoặc thiết bị I/O
	START I/O	Truyền lệnh đến bộ xử lý I/O để bắt đầu hoạt động I/O
	TEST I/O	Truyền thông tin trạng thái từ hệ thống I/O đến đích xác định

f. Điều khiển hệ thống

- Các câu lệnh có thể được thực hiện chỉ khi VXL trong trạng thái đặc quyền hoặc đang thực hiện một chương trình trong vùng đặc quyền đặc biệt của bộ nhớ.
- Thông thường các lệnh này được dành riêng cho hệ điều hành
- Ví dụ về các hoạt động điều khiển hệ thống:
  - Một câu lệnh điều khiển hệ thống có thể đọc hoặc thay đổi thanh ghi điều khiển
  - Câu lệnh để đọc hoặc thay đổi khóa bảo vệ bộ nhớ
  - Truy cập vào các khối điều khiển tiến trình trong hệ thống đa chương trình

g. Truyền điều khiển

- Các hoạt động truyền điều khiển là cần thiết:
  - Cần thiết để có thể thực thi mỗi câu lệnh nhiều hơn một lần
  - Hầu như mọi chương trình đều gồm có việc ra quyết định
  - Cơ chế phân tách các nhiệm vụ ra thành các công việc nhỏ hơn có thể thực hiện tại các thời điểm khác nhau
- Các hoạt động truyền điều khiển nói chung:
  - Rẽ nhánh:
    - Rẽ nhánh có điều kiện: chỉ rẽ nhánh (thiết lập địa chỉ (toán hạng của lệnh) vào thanh ghi PC) khi một điều kiện nhất định được thỏa mãn
    - Rẽ nhánh không điều kiện: việc rẽ nhánh luôn được thực hiện.
  - Lệnh nhảy
  - Lệnh gọi thủ tục

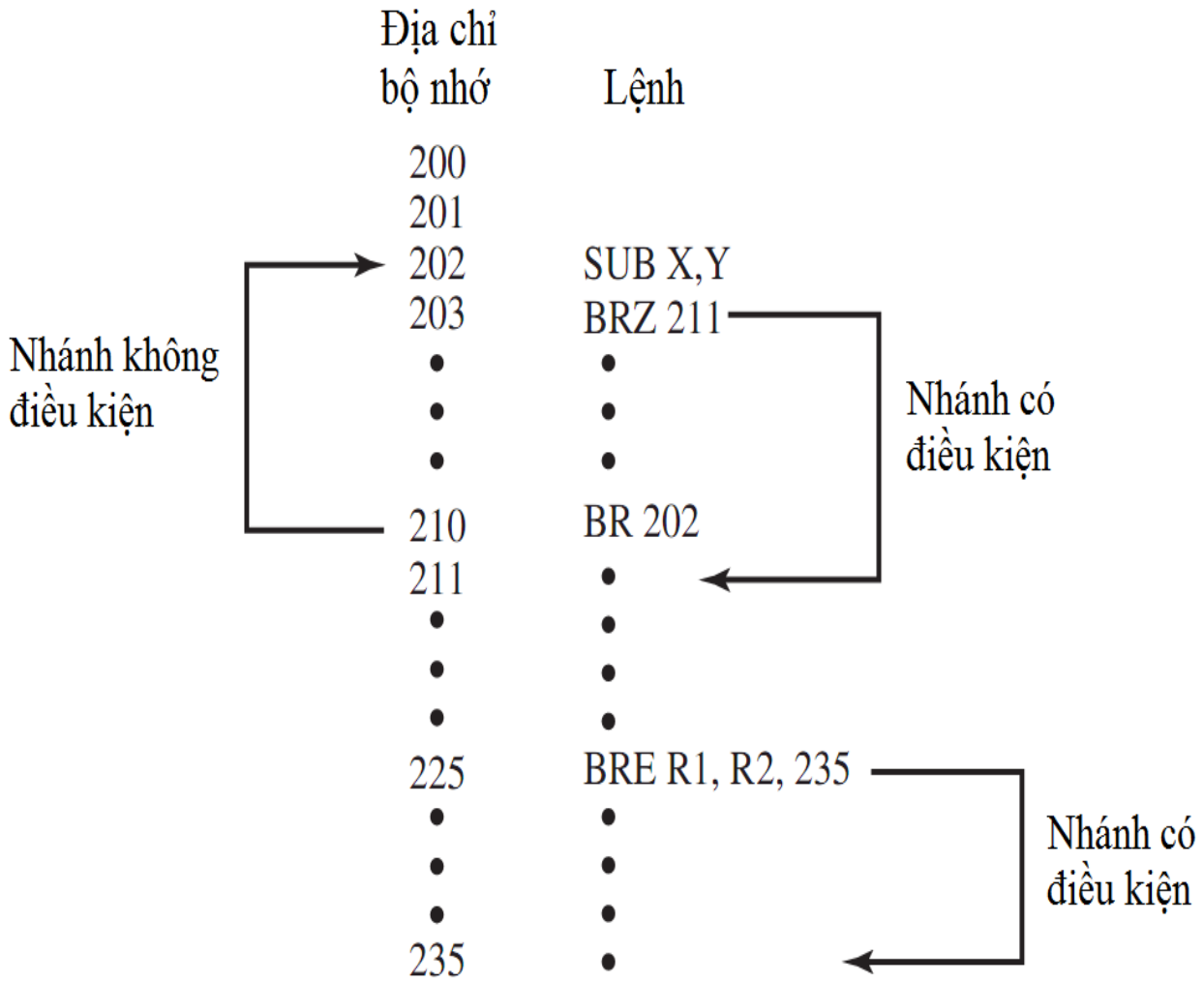
Truyền điều khiển

- Các lệnh nhảy và gọi thủ tục

Truyền điều khiển	JUMP (BRANCH)	Truyền không điều kiện; tải địa chỉ xác định vào PC
	JUMP CONDITIONAL	Kiểm tra điều kiện cụ thể; tải địa chỉ xác định vào PC hoặc không làm gì tùy thuộc vào điều kiện
	CALL	Cất nội dung của PC (địa chỉ trở về) vào một vị trí xác định; Nạp địa chỉ lệnh đầu tiên của chương trình con vào PC
	RETURN	Đặt địa chỉ trở về trả lại cho PC để trở về chương trình chính
	SKIP	Tăng PC để bỏ qua lệnh tiếp theo
	SKIP CONDITIONAL	Kiểm tra điều kiện cụ thể; bỏ qua lệnh hoặc không làm gì tùy thuộc vào điều kiện
	HALT	Dừng thực thi chương trình
	WAIT (HOLD)	Dừng thực thi chương trình; liên tục kiểm tra điều kiện cụ thể; quay lại thực thi tiếp khi điều kiện được thỏa mãn
	NO OPERATION	Không có hành động nào được thực hiện, nhưng việc thực thi chương trình vẫn được tiếp tục

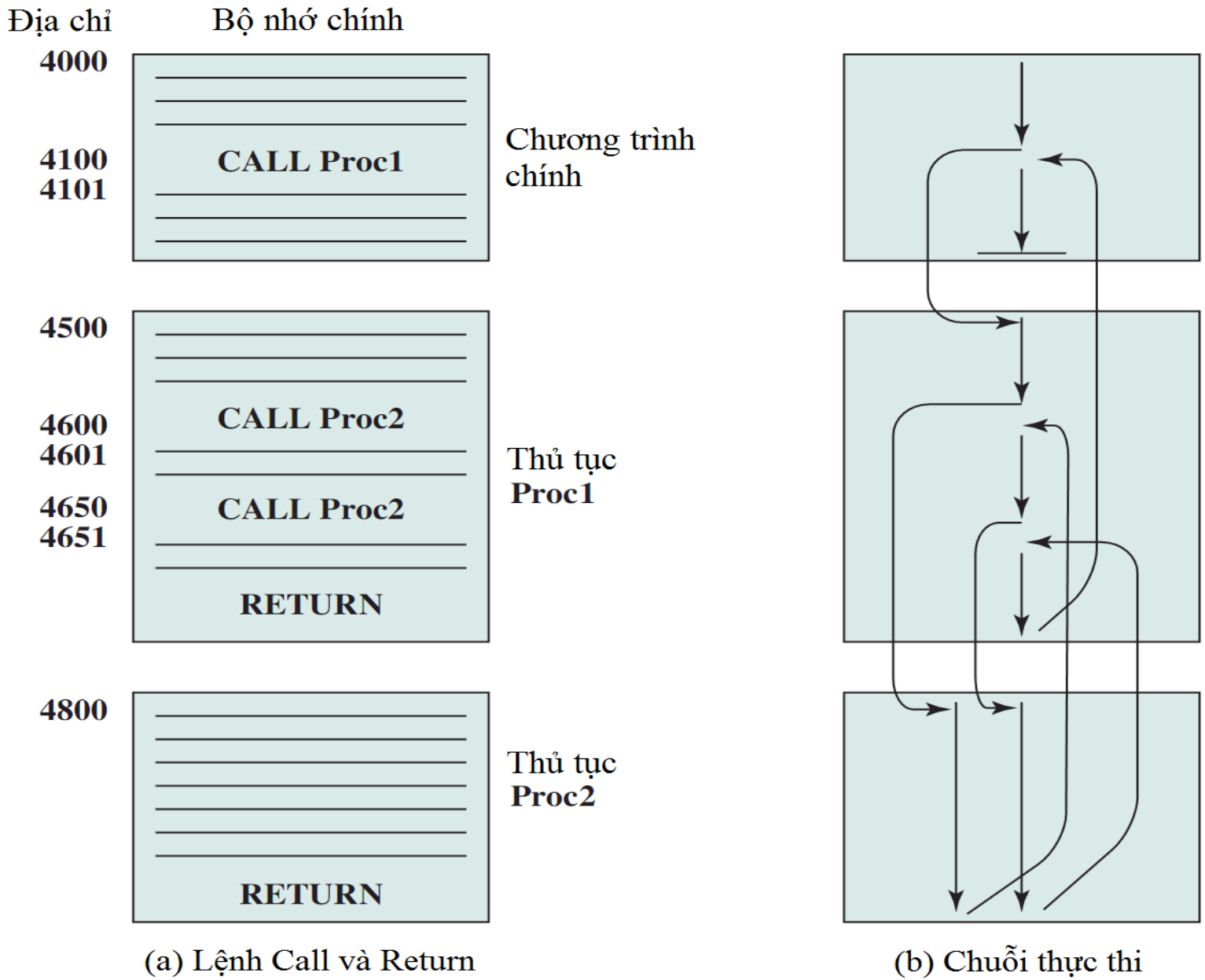
Ví dụ một số lệnh rẽ nhánh

- BRP X: Rẽ nhánh đến vị trí X nếu kết quả dương.
- BRN X: Rẽ nhánh đến vị trí X nếu kết quả âm.
- BRZ X: Rẽ nhánh đến vị trí X nếu kết quả là 0.
- BRO X: Rẽ nhánh đến vị trí X nếu xảy ra tràn.



Lệnh gọi thủ tục

- Thủ tục là một chương trình máy tính được kết hợp vào một chương trình lớn hơn.
  - Thủ tục có thể được gọi tại bất kỳ vị trí nào trong chương trình.
  - Bộ xử lý được chỉ dẫn để đi tới và thực hiện toàn bộ thủ tục rồi sau đó quay trở về điểm phát sinh lệnh gọi thủ tục.
- Cơ chế của thủ tục liên quan đến hai lệnh cơ bản:
  - lệnh gọi thủ tục (call) rẽ nhánh từ vị trí hiện tại đến thủ tục,
  - lệnh trả về (return) để từ thủ tục trở về nơi mà nó được gọi.



(a) Lệnh Call và Return

(b) Chuỗi thực thi

Chương 9. Tập lệnh: Đặc điểm và chức năng

9.1. Đặc điểm tập lệnh

9.2. Các kiểu toán hạng

9.3. Các kiểu dữ liệu Intel x86 và ARM

9.4. Các loại hoạt động

9.5. Các loại hoạt động Intel x86 và ARM

Instruction	Description
Data Movement	
MOV	Move operand, between registers or between register and memory.
PUSH	Push operand onto stack.
PUSHA	Push all registers on stack.
MOVSX	Move byte, word, dword, sign extended. Moves a byte to a word or a word to a doubleword with twos-complement sign extension.
LEA	Load effective address. Loads the offset of the source operand, rather than its value to the destination operand.
XLAT	Table lookup translation. Replaces a byte in AL with a byte from a user-coded translation table. When XLAT is executed, AL should have an unsigned index to the table. XLAT changes the contents of AL from the table index to the table entry.
IN, OUT	Input, output operand from I/O space.
Arithmetic	
ADD	Add operands.
SUB	Subtract operands.
MUL	Unsigned integer multiplication, with byte, word, or double word operands, and word, doubleword, or quadword result.
IDIV	Signed divide.
Logical	
AND	AND operands.
BTS	Bit test and set. Operates on a bit field operand. The instruction copies the current value of a bit to flag CF and sets the original bit to 1.
BSF	Bit scan forward. Scans a word or doubleword for a 1-bit and stores the number of the first 1-bit into a register.
SHL/SHR	Shift logical left or right.
SAL/SAR	Shift arithmetic left or right.
ROL/ROR	Rotate left or right.
SETcc	Sets a byte to zero or one depending on any of the 16 conditions defined by status flags.
Control Transfer	
JMP	Unconditional jump.
CALL	Transfer control to another location. Before transfer, the address of the instruction following the CALL is placed on the stack.
JE/JZ	Jump if equal/zero.
LOOPE/LOOPZ	Loops if equal/zero. This is a conditional jump using a value stored in register ECX. The instruction first decrements ECX before testing ECX for the branch condition.
INT/INTO	Interrupt/Interrupt if overflow. Transfer control to an interrupt service routine

Bảng12.8

Các loại hoạt động x86 (Với các ví dụ của các hoạt động thông thường)

(page 1 of 2)

String Operations	
MOVS	Move byte, word, dword string. The instruction operates on one element of a string, indexed by registers ESI and EDI. After each string operation, the registers are automatically incremented or decremented to point to the next element of the string.
LODS	Load byte, word, dword of string.
High-Level Language Support	
ENTER	Creates a stack frame that can be used to implement the rules of a block-structured high-level language.
LEAVE	Reverses the action of the previous ENTER.
BOUND	Check array bounds. Verifies that the value in operand 1 is within lower and upper limits. The limits are in two adjacent memory locations referenced by operand 2. An interrupt occurs if the value is out of bounds. This instruction is used to check an array index.
Flag Control	
STC	Set Carry flag.
LAHF	Load AH register from flags. Copies SF, ZF, AF, PF, and CF bits into A register.
Segment Register	
LDS	Load pointer into DS and another register.
	System Control
HLT	Halt.
LOCK	Asserts a hold on shared memory so that the Pentium has exclusive use of it during the instruction that immediately follows the LOCK.
ESC	Processor extension escape. An escape code that indicates the succeeding instructions are to be executed by a numeric coprocessor that supports high-precision integer and floating-point calculations.
WAIT	Wait until BUSY# negated. Suspends Pentium program execution until the processor detects that the BUSY pin is inactive, indicating that the numeric coprocessor has finished execution.
Protection	
SGDT	Store global descriptor table.
LSL	Load segment limit. Loads a user-specified register with a segment limit.
VERR/VERW	Verify segment for reading/writing.
Cache Management	
INVD	Flushes the internal cache memory.
WBINVD	Flushes the internal cache memory after writing dirty lines to memory.
INVLPG	Invalidates a translation lookaside buffer (TLB) entry.

Bảng12.8

Các loại hoạt động x86 (Với các ví dụ của các hoạt động thông thường)

(page 2 of 2)

Các cờ trạng thái x86

Status Bit	Name	Description
CF	Carry	Indicates carrying or borrowing out of the left-most bit position following an arithmetic operation. Also modified by some of the shift and rotate operations.
PF	Parity	Parity of the least-significant byte of the result of an arithmetic or logic operation. 1 indicates even parity; 0 indicates odd parity.
AF	Auxiliary Carry	Represents carrying or borrowing between half-bytes of an 8-bit arithmetic or logic operation. Used in binary-coded decimal arithmetic.
ZF	Zero	Indicates that the result of an arithmetic or logic operation is 0.
SF	Sign	Indicates the sign of the result of an arithmetic or logic operation.
OF	Overflow	Indicates an arithmetic overflow after an addition or subtraction for twos complement arithmetic.

Symbol	Condition Tested	Comment
A, NBE	CF=0 AND ZF=0	Above; Not below or equal (greater than, unsigned)
AE, NB, NC	CF=0	Above or equal; Not below (greater than or equal, unsigned); Not carry
B, NAE, C	CF=1	Below; Not above or equal (less than, unsigned); Carry set
BE, NA	CF=1 OR ZF=1	Below or equal; Not above (less than or equal, unsigned)
E, Z	ZF=1	Equal; Zero (signed or unsigned)
G, NLE	[(SF=1 AND OF=1) OR (SF=0 and OF=0)] AND [ZF=0]	Greater than; Not less than or equal (signed)
GE, NL	(SF=1 AND OF=1) OR (SF=0 AND OF=0)	Greater than or equal; Not less than (signed)
L, NGE	(SF=1 AND OF=0) OR (SF=0 AND OF=1)	Less than; Not greater than or equal (signed)
LE, NG	(SF=1 AND OF=0) OR (SF=0 AND OF=1) OR (ZF=1)	Less than or equal; Not greater than (signed)
NE, NZ	ZF=0	Not equal; Not zero (signed or unsigned)
NO	OF=0	No overflow
NS	SF=0	Not sign (not negative)
NP, PO	PF=0	Not parity; Parity odd
O	OF=1	Overflow
P	PF=1	Parity; Parity even
S	SF=1	Sign (negative)

Bảng 12.10

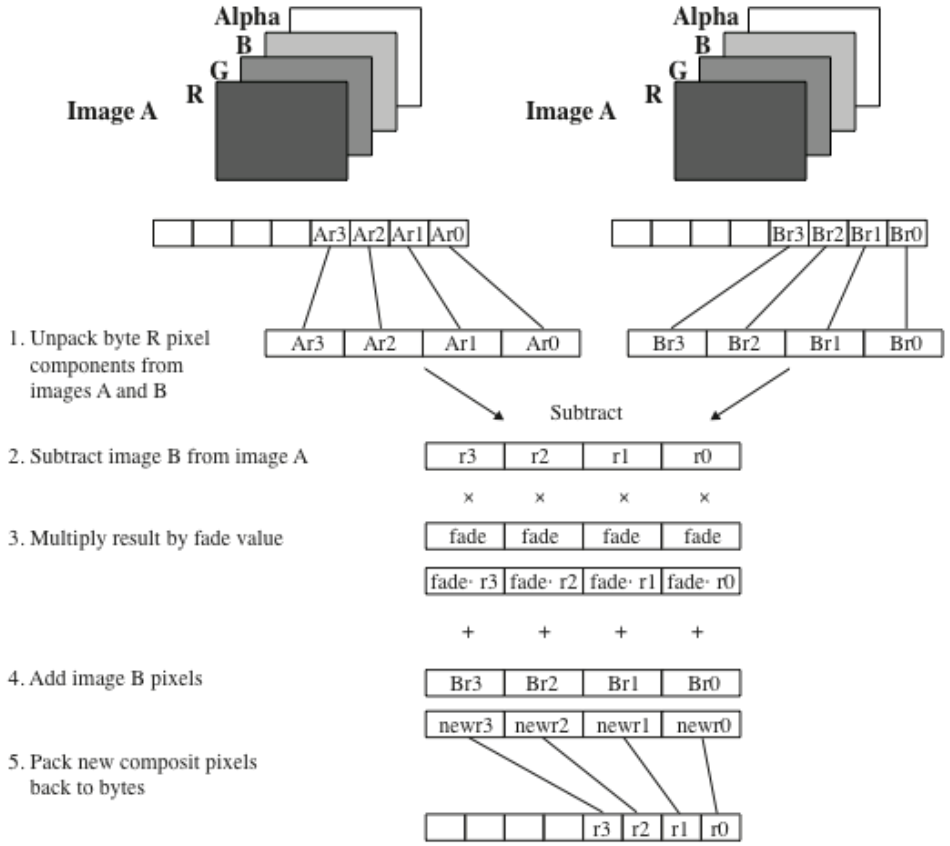
Các mã điều kiện x86 cho lệnh SETcc và lệnh nhảy có điều kiện

Tập lệnh MMX

Category	Instruction	Description
Arithmetic	PADD [B, W, D]	Parallel add of packed eight bytes, four 16-bit words, or two 32-bit doublewords, with wraparound.
	PADDS [B, W]	Add with saturation.
	PADDUS [B, W]	Add unsigned with saturation
	PSUB [B, W, D]	Subtract with wraparound.
	PSUBS [B, W]	Subtract with saturation.
	PSUBUS [B, W]	Subtract unsigned with saturation
	PMULHW	Parallel multiply of four signed 16-bit words, with high-order 16 bits of 32-bit result chosen.
	PMULLW	Parallel multiply of four signed 16-bit words, with low-order 16 bits of 32-bit result chosen.
	PMADDWD	Parallel multiply of four signed 16-bit words; add together adjacent pairs of 32-bit results.
Comparison	PCMPEQ [B, W, D]	Parallel compare for equality; result is mask of 1s if true or 0s if false.
	PCMPGT [B, W, D]	Parallel compare for greater than; result is mask of 1s if true or 0s if false.
Conversion	PACKUSWB	Pack words into bytes with unsigned saturation.
	PACKSS [WB, DW]	Pack words into bytes, or doublewords into words, with signed saturation.
	PUNPCKH [BW, WD, DQ]	Parallel unpack (interleaved merge) high-order bytes, words, or doublewords from MMX register.
	PUNPCKL [BW, WD, DQ]	Parallel unpack (interleaved merge) low-order bytes, words, or doublewords from MMX register.
Logical	PAND	64-bit bitwise logical AND
	PNDN	64-bit bitwise logical AND NOT
	POR	64-bit bitwise logical OR
	PXOR	64-bit bitwise logical XOR
Shift	PSLL [W, D, Q]	Parallel logical left shift of packed words, doublewords, or quadword by amount specified in MMX register or immediate value.
	PSRL [W, D, Q]	Parallel logical right shift of packed words, doublewords, or quadword.
	PSRA [W, D]	Parallel arithmetic right shift of packed words, doublewords, or quadword.
Data Transfer	MOV [D, Q]	Move doubleword or quadword to/from MMX register.
State Mgt	EMMS	Empty MMX state (empty FP registers tag bits).

Note: If an instruction supports multiple data types [byte (B), word (W), doubleword (D), quadword (Q)], the data types are indicated in brackets.

Image Compositing on Color Plane Representation



MMX code sequence performing this operation:

```
pxor      mm7, mm7      ;zero out mm7
movq      mm3, fad_val  ;load fade value replicated 4 times
movd      mm0, imageA   ;load 4 red pixel components from image A
movd      mm1, imageB   ;load 4 red pixel components from image B
punpckblw mm0, mm7      ;unpack 4 pixels to 16 bits
punpckblw mm1, mm7      ;unpack 4 pixels to 16 bits
psubw     mm0, mm1      ;subtract image B from image A
pmulhw    mm0, mm3      ;multiply the subtract result by fade values
padddw    mm0, mm1      ;add result to image B
packuswb  mm0, mm7      ;pack 16-bit results back to bytes
```

Figure 12.11 Image Compositing on Color Plane Representation

Các loại câu lệnh ARM

Các lệnh tải và lưu trữ dữ liệu

Các lệnh rẽ nhánh

Các lệnh xử lý dữ liệu

Các lệnh nhân

Các lệnh cộng trừ song song

Các lệnh mở rộng

Các lệnh truy cập vào thanh ghi trạng thái

Code	Symbol	Condition Tested	Comment
0000	EQ	Z = 1	Equal
0001	NE	Z = 0	Not equal
0010	CS/HS	C = 1	Carry set/unsigned higher or same
0011	CC/LO	C = 0	Carry clear/unsigned lower
0100	MI	N = 1	Minus/negative
0101	PL	N = 0	Plus/positive or zero
0110	VS	V = 1	Overflow
0111	VC	V = 0	No overflow
1000	HI	C = 1 AND Z = 0	Unsigned higher
1001	LS	C = 0 OR Z = 1	Unsigned lower or same
1010	GE	N = V [(N = 1 AND V = 1) OR (N = 0 AND V = 0)	Signed greater than or equal
1011	LT	N ≠ V [(N = 1 AND V = 0) OR (N = 0 AND V = 1)]	Signed less than
1100	GT	(Z = 0) AND (N = V)	Signed greater than
1101	LE	(Z = 1) OR (N ≠ V)	Signed less than or equal
1110	AL	—	Always (unconditional)
1111	—	—	This instruction can only be executed unconditionally

Các điều kiện ARM cho thực thi câu lệnh điều kiện

Tổng kết

Chương 9

- Đặc điểm lệnh máy
  - Các thành phần của lệnh máy
  - Biểu diễn lệnh
  - Các loại lệnh
  - Số lượng địa chỉ
  - Thiết kế tập lệnh
- Các loại toán hạng
  - Số
  - Ký tự
  - Dữ liệu logic

Tập lệnh:

Đặc điểm và chức năng

- Các kiểu dữ liệu Intel x86 và ARM
- Các loại hoạt động
  - Truyền dữ liệu
  - Tính toán số học
  - Logical
  - Chuyển đổi
  - Vào/ra
  - Điều khiển hệ thống
  - Truyền điều khiển
- Các loại hoạt động trong Intel x86 và ARM

Câu hỏi ôn tập

1. Các thành phần điển hình của một lệnh máy?
2. Toán hạng nguồn và đích có thể được đặt ở đâu?
3. Trình bày ngắn gọn 5 vấn đề quan trọng của thiết kế tập lệnh
4. Các loại toán hạng điển hình trong một tập lệnh máy là gì?
5. Liệt kê các loại lệnh điển hình của máy tính.
6. Phân biệt phép dịch số học và dịch logic.
7. Tại sao cần phải có lệnh chuyển điều khiển?

Hình ảnh và nội dung trong bài giảng này tham khảo từ cuốn sách và slide bài giảng “Computer Organization and Architecture”, 10th Edition, của tác giả William Stallings.