VIETNAM NATIONAL UNIVERSITY HANOI (VNU)
VNU UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Computer Architecture

Lecture 9: Instruction Set Architecture

Duy-Hieu Bui, PhD
AIoT Laboratory
VNU Information Technology Institute
Email: hieubd@vnu.edu.vn
https://duyhieubui.github.io

Nội dung

- Khái niệm
- Biểu diễn lệnh
- Format lệnh
- Các kiểu đánh địa chỉ

Tham khảo chương 12, 13 của "Computer Organization and Architecture: Designing for Performance", William Stallings, 10th edition

1. Khái niệm

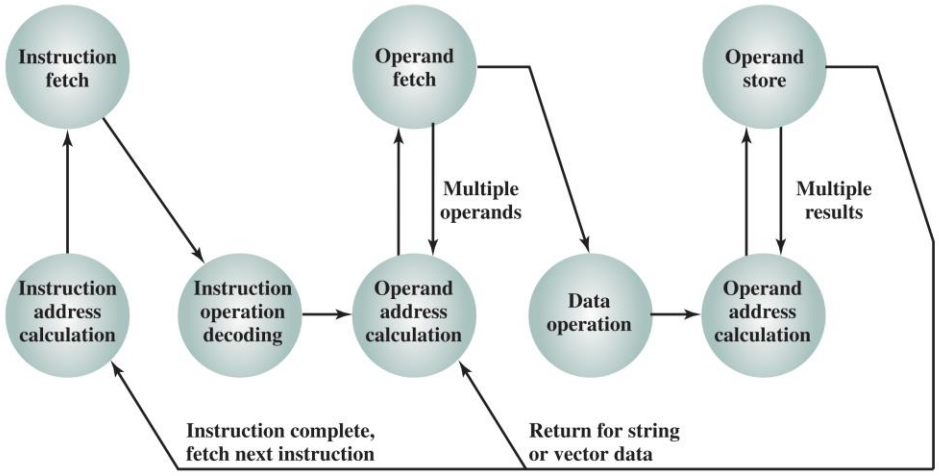
- Tập lệnh: tập đầy đủ các lệnh mà CPU hiểu được.
  - Lệnh: Mã máy (binary), thường được biểu diễn bởi những mã hợp ngữ (assembly codes)
  - Phần nhìn thấy của máy tính bởi người lập trình (đặc biệt đối với người viết chương trình dịch)
  - Thể hiện khái quát về mặt logic một máy tính theo nghĩa các registers, hoạt động của ALU, kiểu dữ liệu, …
  - Thiết kế tập lệnh là một phần quan trọng trong việc thiết kế CPU
  - Mỗi một kiểu máy tính có một tập lệnh đặc thù.

Khái niệm…

Operator(Src_1, Src_2, …, Src_n) ➔ (Dst_1, Dst_2, …, Dst_m)

- Một lệnh phải chứa những thông tin đòi hỏi bởi CPU:
  - Mã lệnh (operation code – opcode): mã nhị phân xác định thao tác phải thi hành
  - Tham chiếu đến các toán hạng nguồn
  - Tham chiếu đến toán hạng đích
  - Tham chiếu đến lệnh kế tiếp
- Các toán hạng có thể là:
  1. Giá trị đưa vào – immediate
  2. Nội dung trong thanh ghi
  3. Nội dung từ nhớ trong bộ nhớ chính/bộ nhớ ảo
  4. Nội dung từ nhớ trên thiết bị I/O

Sơ đồ trạng thái chu trình lệnh



2. Biểu diễn lệnh

- Biểu diễn lệnh: chuỗi các bits được chia thành các trường

- Biểu diễn tượng trưng (mnemonics): cả opcode lẫn các toán hạng
  - ADD Add
  - SUB Subtract
  - MUL Multiply
  - DIV Divide
  - LOAD Load data from memory
  - STORE Store data to memory

Ví dụ

Địa chỉ bộ nhớ	Nội dung				Diễn dịch
0100	0010	0010	0000	1100	LOAD (1100)
0101	0001	0010	0000	1101	ADD (1101)
0110	0001	0010	0000	1110	ADD (1110)
0111	0011	0010	0000	1111	STORE (1111)
1100	0000	0000	0000	0010	0002
1101	0000	0000	0000	0011	0003
1110	0000	0000	0000	0100	0004
1111	0000	0000	0000	0000	0000

Ngôn ngữ máy tính

- Được chia làm nhiều bậc khác nhau:
  - Bậc thấp LLL: ngôn ngữ máy (binary), hợp ngữ…
  - Bậc cao HLL: C, Pascal, Basic
- Một lệnh HLL tương ứng với nhiều lệnh LLL
- **Tập lệnh phải đảm bảo đủ khả năng mã hoá tất cả các lệnh của một ngôn ngữ bậc cao**.

Ví dụ : X = X + Y được dịch thành:

- LOAD X, R1
- ADD R1, Y
- STORE R1, X

Thiết kế tập lệnh

Thoả hiệp giữa:

- Số lượng phép toán
- Độ phức tạp của các phép toán
- Số kiểu dữ liệu
- Số thanh ghi registers
- Phương thức sử dụng registers
- Các kiểu đánh địa chỉ
- Số lượng trường trong một lệnh
- Độ lớn của các trường

Thiết kế tập lệnh → thiết kế CPU

3. Format lệnh

- Phân loại tập lệnh theo format lệnh: dựa trên số lượng địa chỉ toán hạng tham chiếu
  - Lý thuyết: cần 4 trường để chứa địa chỉ
    - Toán hạng nguồn 1
    - Toán hạng nguồn 2
    - Toán hạng kết quả
    - Lệnh kế tiếp
  - Thực tế:
    - 3 địa chỉ: ít sử dụng
    - 2 địa chỉ: 1 cho nguồn và 1 cho đích
    - 1 địa chỉ: sử dụng accumulator để chứa một toán hạng và kết quả
    - 0 địa chỉ: sử dụng một stack để chứa các toán hạng và kết quả

Format lệnh…

Số địa chỉ	Biểu diễn	Nội dung
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	ACC ← ACC OP A
0	OP	T ← (T-1) OP T

ACC : accumulation register (accumulator)

T: đỉnh của stack (LIFO)

- Ảnh hưởng việc chọn số địa chỉ :
  - Càng ít số địa chỉ, lệnh càng ngắn hơn
  - CPU càng ít phức tạp hơn,
  - càng nhiều số lệnh và các chương trình thi hành sẽ chậm hơn

Hiện tại: kết hợp formats 2 địa chỉ và 3 địa chỉ

Ví dụ

Instruction		Comment
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) Three-address instructions

Instruction		Comment
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) Two-address instructions

- Tính $Y = \frac{A-B}{C+(D\times E)}$

Instruction		Comment
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) One-address instructions

Quan hệ memory-register

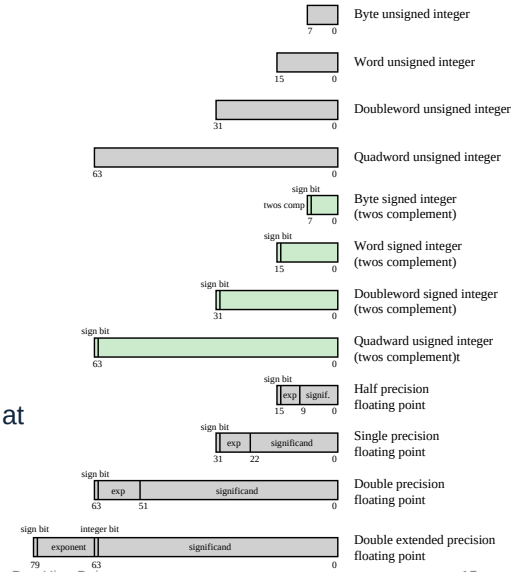
- Registers: thành phần nhớ có tốc độ truy cập/ghi nhanh trong CPU và làm giảm thiểu tần xuất truy cập bộ nhớ
- Các chiến lược thao tác dữ liệu:
  - register-register:
    - LOAD và STORE thực hiện tương tác với bộ nhớ
    - Lệnh đơn giản, thi hành nhanh và số lượng lệnh sinh tương đối lớn
  - register-memory:
    - Mã sinh ra gọn
    - Khó giải mã lệnh và không cố định số chu trình khi thi hành
  - memory-memory:
    - Truy cập trực tiếp đến bộ nhớ => có thể dẫn đến tình trạng nghẽn

Phân loại toán hạng

- Địa chỉ: số nguyên không dấu
- Số: nguyên, thực, BCD, …
- Ký tự: ASCII, Unicode, …
- Dữ liệu logic: bits, flag,

Kiểu dữ liệu của x86 - Intel

- 8 bit Byte
- 16 bit word
- 32 bit double word
- 64 bit quad word
- Addressing is by 8 bit unit
- A 32 bit double word is read at addresses divisible by 4



Kiểu dữ liệu của PowerPC

- Độ lớn: 8 (byte), 16 (halfword), 32 (word) và 64 (doubleword)
- Một số lệnh cần toán hạng quy về giới hạn 32 bits
- Fixed point:
  - Unsigned byte, unsigned halfword, signed halfword, unsigned word, signed word, unsigned doubleword, byte string (<128 bytes)
- Floating point: IEEE 754
  - Single, or
  - double precision

Phân loại lệnh

- Xử lý dữ liệu:
  - Thao tác số học: số nguyên, số thực
  - Logic
  - Chuyển đổi
- Chuyển dữ liệu (I/O)
- Lưu dữ liệu (main memory)
- Điều khiển:
  - Kiểm tra và rẽ nhánh
  - Kiểm tra các thanh ghi điều kiện

Một số lệnh thông dụng

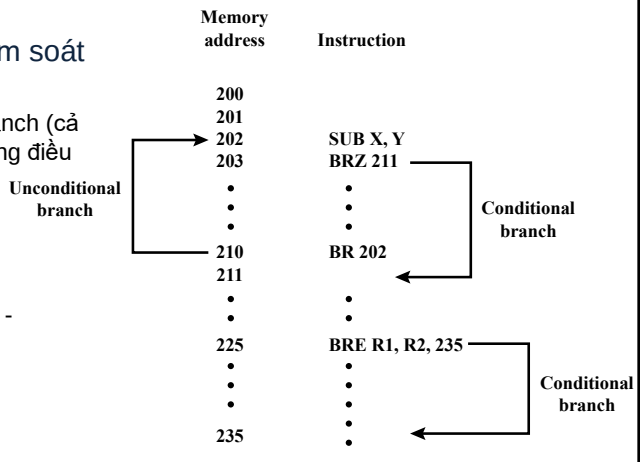
Type	Operation Name	Description
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end

Một số lệnh thông dụng …

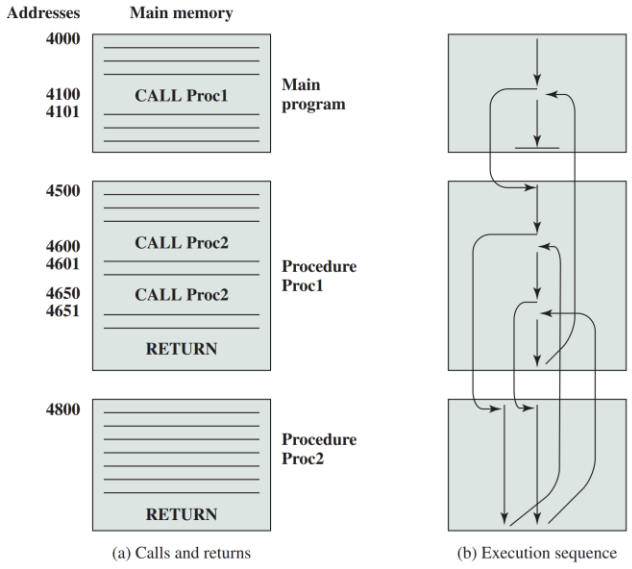
Type	Operation Name	Description
Transfer of Control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued
Input/Output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

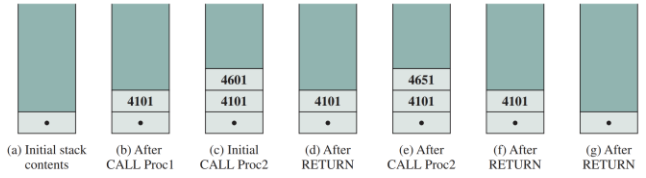
Transfer of Control

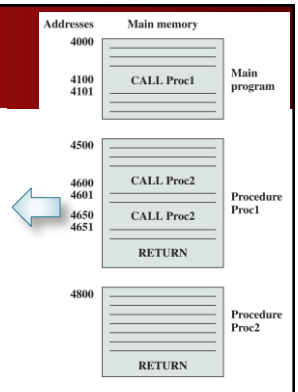
- Chương trình cần phải được kiểm soát!
- Các lệnh hỗ trợ kiểm soát gồm:
  - Lệnh rẽ nhánh – Branch (cả có điều kiện lẫn không điều kiện)
  - Lệnh nhảy - Skip
  - Lệnh gọi thủ tục con - Procedure call



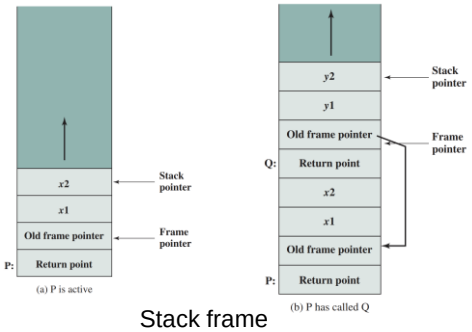
Lệnh gọi thủ tục …







The use of stack for subroutine/function call

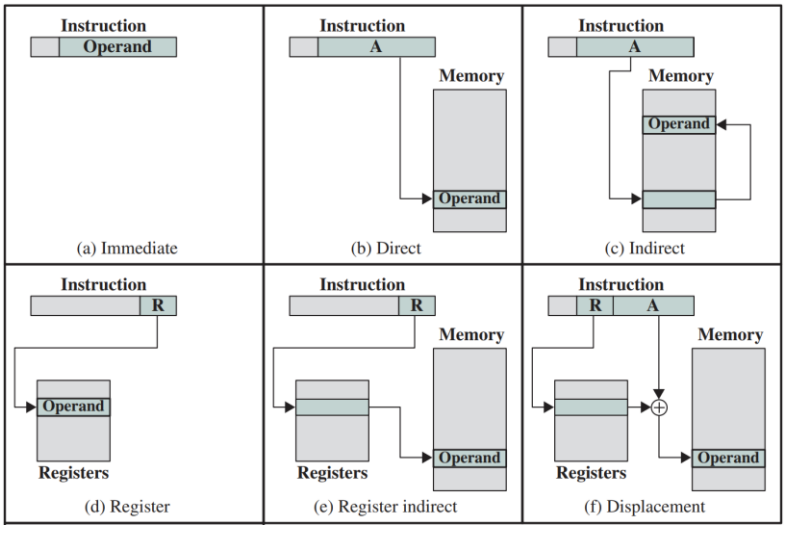


Stack frame

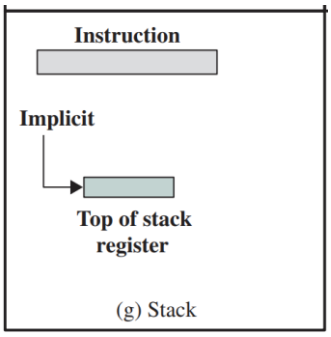
4. Kiểu đánh địa chỉ

- Tức thời - immediate:
  - Không cần tham chiếu đến bộ nhớ,
  - Độ lớn của toán hạng bị giới hạn.
- Trực tiếp :
  - Đơn giản,
  - Độ lớn không gian địa chỉ bị giới hạn.
- Thanh ghi:
  - Không cần tham chiếu đến bộ nhớ,
  - Độ lớn không gian địa chỉ bị giới hạn.
- Gián tiếp qua bộ nhớ:
  - Nhiều tham chiếu đến bộ nhớ
- Gián tiếp qua thanh ghi
- Dịch chuyển:
  - Mềm dẻo
  - Phức tạp

Kiểu đánh địa chỉ



Kiểu đánh địa chỉ



Minh hoạ

register	ADD R4, R3
immediate	ADD R4, 3
direct	ADD R4, (0011)
indirect by register	ADD R4, (R3)
indirect by memory	ADD R4, @(0011)
displacement	ADD R4, (R3)100

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

Đánh địa chỉ

- Sự đối kháng giữa
  - Không gian có thể đánh được địa chỉ và tính linh hoạt
  - Số tham chiếu bộ nhớ và độ phức tạp của việc tính toán địa chỉ
- Có nhiều kiểu đánh địa chỉ khác nhau trong máy tính
- Các kiểu immediate, indirect by register và displacement thường được sử dụng nhiều nhất
- 2 cách phân biệt kiểu đánh địa chỉ:
  - Sử dụng một hay nhiều bits (address *specificator*)
    - Cần thiết khi có một số lượng lớn kiểu
    - Có thể dẫn đến độ dài lệnh thay đổi
  - Sử dụng mã lệnh opcodes khác nhau:
    - Cho phép bảo đảm kích thước lệnh cố định
    - Đơn giản hơn cho phần cứng

Độ lớn lệnh

- Tập lệnh càng phức tạp thì:
  - Số lượng lệnh trong một chương trình càng giảm,
  - Càng làm tăng độ dài lệnh,
  - Càng sử dụng nhiều không gian nhớ hơn.
- Độ dài một lệnh phụ thuộc:
  - Kích thước và tổ chức bộ nhớ
  - Cấu trúc hệ thống liên kết (bus)
  - Độ phức tạp và tốc độ của CPU
- Kích thước lệnh có thể :
  - cố định
  - thay đổi:
    - Cho phép một thang mã lệnh rộng (có kích thước khác nhau)
    - Tăng tính linh hoạt cho việc đánh địa chỉ
    - Tăng độ phức tạp của CPU

Cấp phát bits

Việc phân chia các trường trong một lệnh phụ thuộc:

- Số kiểu đánh địa chỉ
- Số toán hạng
- Việc sử dụng register
- Số lượng tập register
- Miền địa chỉ (không gian có thể đánh địa chỉ được)
- Phương thức đánh địa chỉ bộ nhớ

- Nếu muốn có đồng thời:
  - Kích thước lệnh hợp lý
  - Khả năng đánh địa chỉ hợp lý
  - Số lượng lớn opcodes

ta có thể sử dụng opcode có kích thước thay đổi

Mã lệnh mở rộng

40 opcodes trong đó chỉ cần 15 lệnh có tham số 12 bit

opcode 4 bits	parameters 12 bits

opcode 4 bits	Extensive opcode 5 bits	Params 7 bits

Chỉ cần 16 bits thay vì 18 bits !

Ví dụ: ALPHA - DEC

- 32 registers - 64 bits : thao tác với số nguyên
- 32 registers - 64 bits : thao tác với số thực
- Kích thước lệnh cố định (32 bits)
- 4 formats lệnh:
  a. instructions riêng cho OS
  b. Rẽ nhánh
  c. Chuyển đổi dữ liệu
  d. Tính toán số tự nhiên hoặc thực

ALPHA

a

Opcode 6 bits	Number 26 bits

b

Opcode 6 bits	Ra 5 bits	Displacement 21 bits

c

Opcode 6 bits	Ra 5 bits	Rb 5 bits	Displacement 16 bits

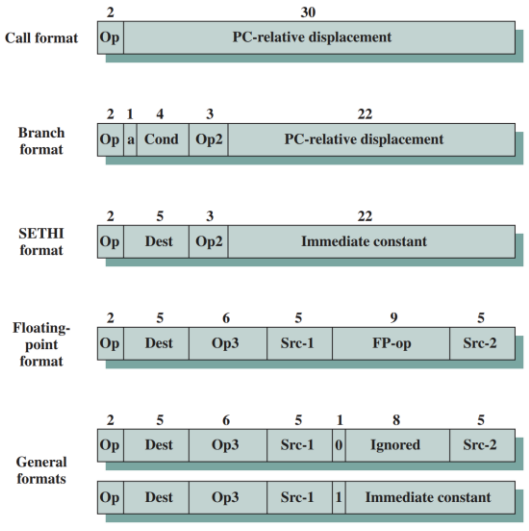
d

Opcode 6 bits	Ra 5 bits	Rb 5 bits	Fonction 11 bits	Displacement 5 bits

Ví dụ: SPARC - SUN

- Số lượng thanh ghi lớn (> 100)
- Có thể truy cập đồng thời 32 registers
- 4 nhóm registers riêng biệt
- Kích thước lệnh cố định (32 bits)
- 3 formats lệnh (format được mã hoá = 2) :
  - Gọi chương trình con
  - Rẽ nhánh hoặc nạp dữ liệu lên register
  - Các thao tác khác với format 3 địa chỉ.

SPARC : formats lệnh



PowerPC : Kiểu đánh địa chỉ

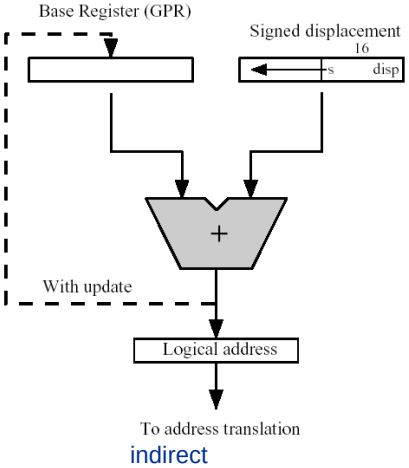
EA = effective address
(X) = contents of X
BR = base register
IR = index register
L/CR = link or count register
GPR = general purpose register
FPR = floating point register
D = displacement
I = immediate value
PC = program counter

Mode	Algorithme
	Load/Store
indirect	EA = (BR) + D
indirect indexed	EA = (BR) + (IR)
	Branch
absolu	EA = I
Relative	EA = (PC) + I
Indirect	EA = (L/CR)
	Integer calculation
register	EA = GPR
immediate	opérande = I
	Floating calculation
register	EA = FPR

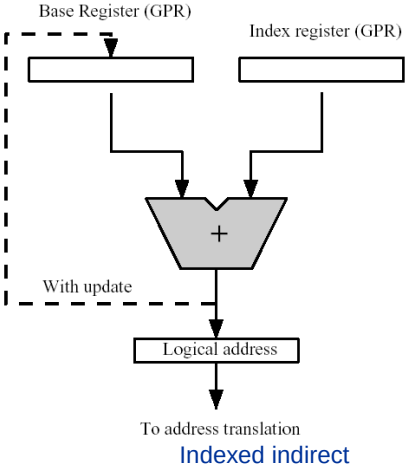
PowerPC…

Các kiểu đánh địa chỉ được phân theo format lệnh :

- Load/Store: indirect và indexed indirect



indirect



Indexed indirect

INTEL – x86

- 8 general registers 32 bits
- 8 general registers 16 bits
- 8 general registers 8 bits

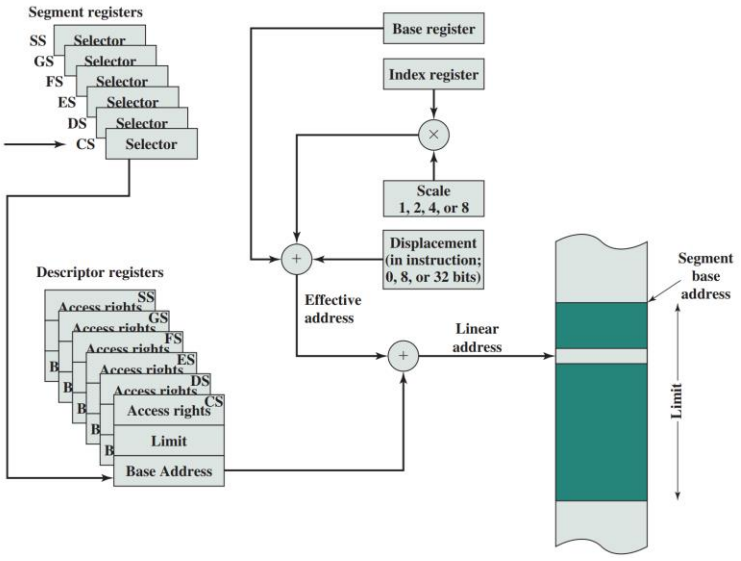
2 registers 32 bits được sử dụng cho các toán hạng 64 bits

- address = segment + offset
  - 6 segment registers
  - 6 descriptor registers
  - 1 base register
  - 1 index register
- Chi tiết tham khảo tại địa chỉ
  https://en.wikipedia.org/wiki/X86_instruction_listings

x86 - INTEL: các kiểu đánh địa chỉ

- immediate
- register
- displacement
  content of segment register + displacement
- indirect by register
  content of segment register + content of base register
- base and displacement
  content of segment register + content of base register + displacement contained in the instruction
- base + index + displacement
  Ditto previous + content of index register
- base + scaled index + displacement
  Ditto previous + content of index register X scaled factor *(scaled factor: 1, 2, 4, 8)*
- scaled index + displacement
  content of segment register + content of base register X scaled factor + displacement contained in the instruction
- relative
  content of PC + displacement contained in the instruction

Address calculation

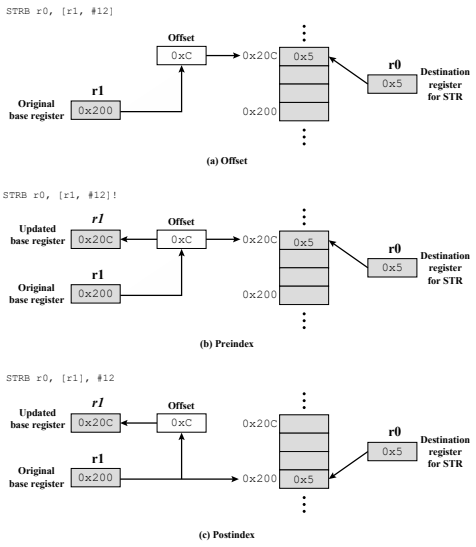


Các kiểu địa chỉ của x86

Mode	Algorithm
Immediate	Operand = A
Register Operand	LA = R
Displacement	LA = (SR) + A
Base	LA = (SR) + (B)
Base with Displacement	LA = (SR) + (B) + A
Scaled Index with Displacement	LA = (SR) + (I) ´ S + A
Base with Index and Displacement	LA = (SR) + (B) + (I) + A
Base with Scaled Index and Displacement	LA = (SR) + (I) ´ S + (B) + A
Relative	LA = (PC) + A

LA = linear address
(X) = contents of X
SR = segment register
PC = program counter
A = contents of an address field in the instruction
R = register
B = base register
I = index register
S = scaling factor

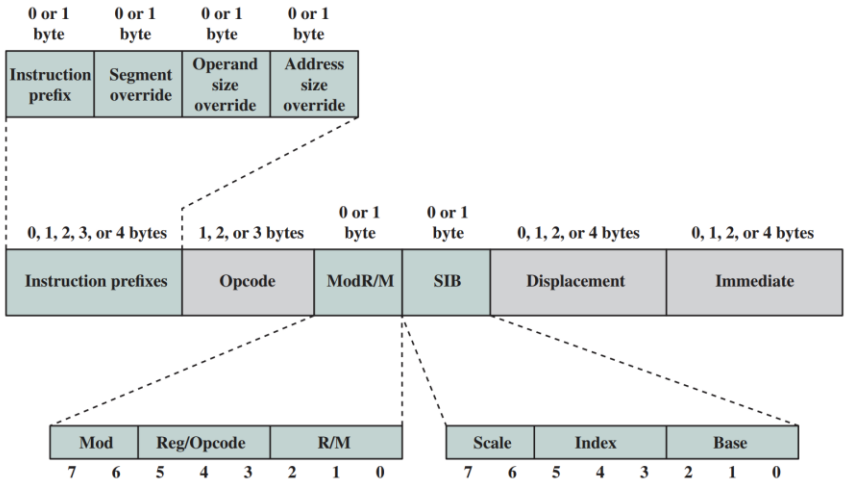
ARM Indexing Methods



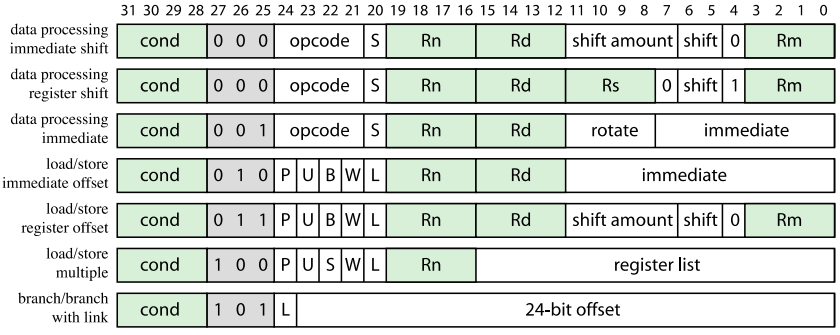
Format lệnh của x86

- Nhiều kiểu lệnh khác nhau:
  - Hiệu quả khi thi hành các lệnh thể hiện bởi các ngôn ngữ bậc cao
  - Tuân thủ sự tương thích trong dòng 8086
- Kiểu đánh địa chỉ được xác định thông qua opcode
- Một lệnh bao gồm:
  - Prefix: 0, 1, 2, 3 hoặc 4 bytes,
  - Opcode: 1 or 2 bytes,
  - Address specificator: 0, 1 or 2 bytes,
  - Displacement: 0, 1, 2 or 4 bytes,
  - Immediate: 0, 1, 2 or 4 bytes

Format lệnh của x86



Format lệnh trong ARM



S = For data processing instructions, signifies that the instruction updates the condition codes
S = For load/store multiple instructions, signifies whether instruction execution is restricted to supervisor mode
P, U, W = bits that distinguish among different types of addressing_mode
B = Distinguishes between an unsigned byte (B==1) and a word (B==0) access
L = For load/store instructions, distinguishes between a Load (L==1) and a Store (L==0)
L = For branch instructions, determines whether a return address is stored in the link register

Tổng kết

- Khái niệm tập lệnh, kiểu lệnh, format lệnh
- Các yếu tố cơ bản quyết định đến tập lệnh
- Các hình thức tham chiếu, đánh địa chỉ trong tập lệnh
  - Immediate, Direct, Indirect, Register, Register Indirect, Displacement (Indexed), Stack
- Các format lệnh trong một số CPU
  - Format lệnh của Intel
  - Format lệnh của SUN
  - Format lệnh của PowerPC

Bài tập

So sánh 4 kiểu kiến trúc lệnh sử dụng
- accumulator
- memory-memory
- stack
- load-store

Giả thiết:
- 1 byte: opcode
- 2 bytes: địa chỉ nhớ
- 4 bytes: toán hạng
- Kích thước lệnh là hệ số của 1 byte