Kiến trúc máy tính

Tập lệnh

NGUYỄN Ngọc Hoá

Bộ môn Hệ thống thông tin, Khoa CNTT

Trường Đại học Công nghệ,

Đại học Quốc gia Hà Nội

Hoa.Nguyen@vnu.edu.vn

Nội dung

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

Tham khảo chương 10, 11 của [1]

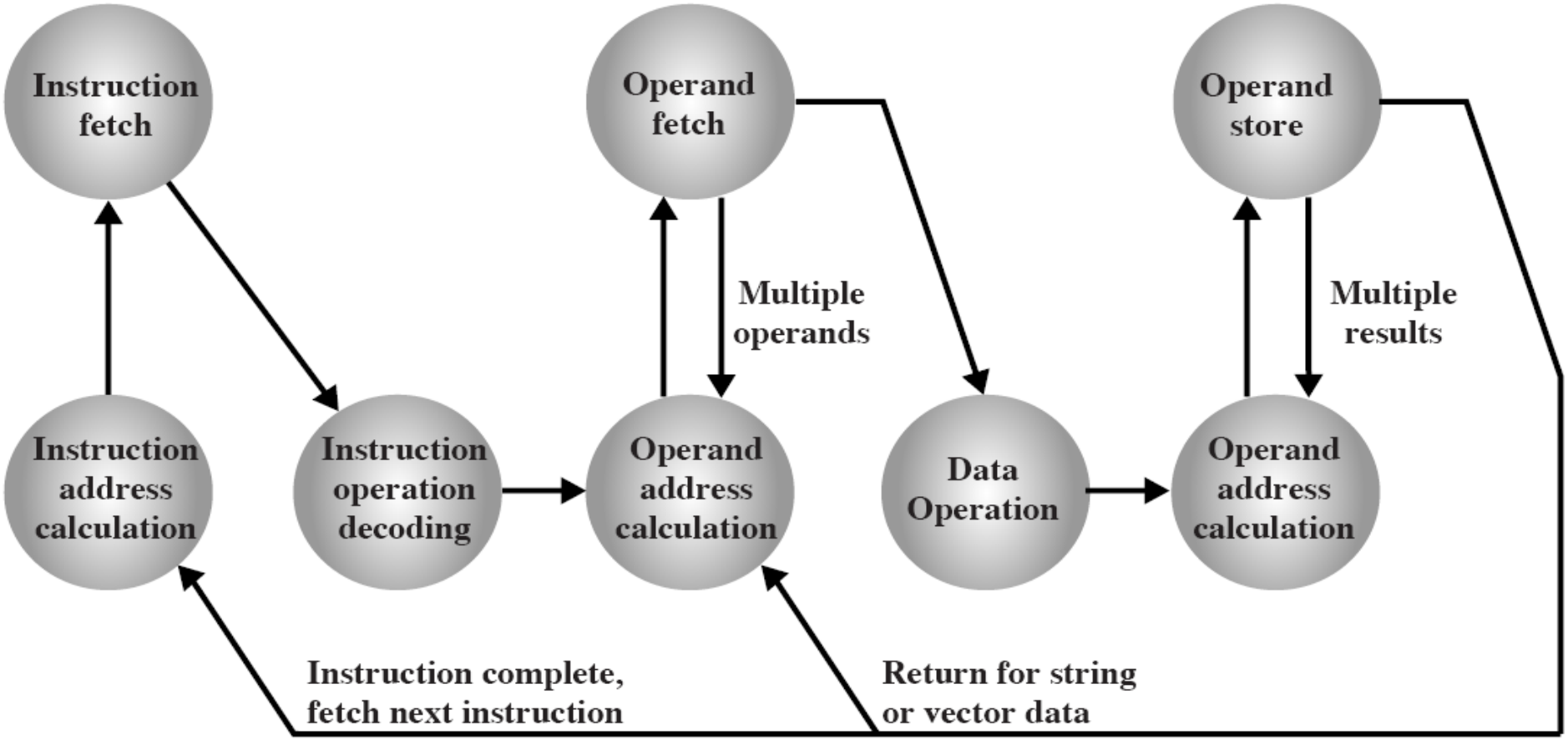
1. Khái niệm

- Tập lệnh: tập đầy đủ các lệnh mà CPU hiểu được.
  - Lệnh: Mã máy (binary), thường được biểu diễn bởi những mã hợp lệnh (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 và một CPU đặc thù.

Khái niệm...

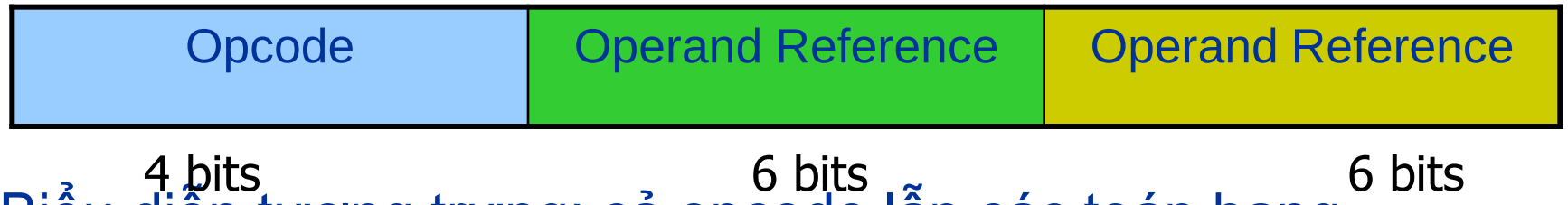
- Một lệnh phải chứa những thông tin đòi hỏi bởi CPU:
  - Mã lệnh (operation code – opcode): mã nhị phân xác định thao tác phải thi hành
  - Tham chiếu đến các toán hạng nguồn
  - Tham chiếu đến toán hạng đích
  - Tham chiếu đến lệnh kế tiếp

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: cả opcode lẫn các toán hạng

Ex: ADD A,B

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 OP T - 1

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

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 lệnh

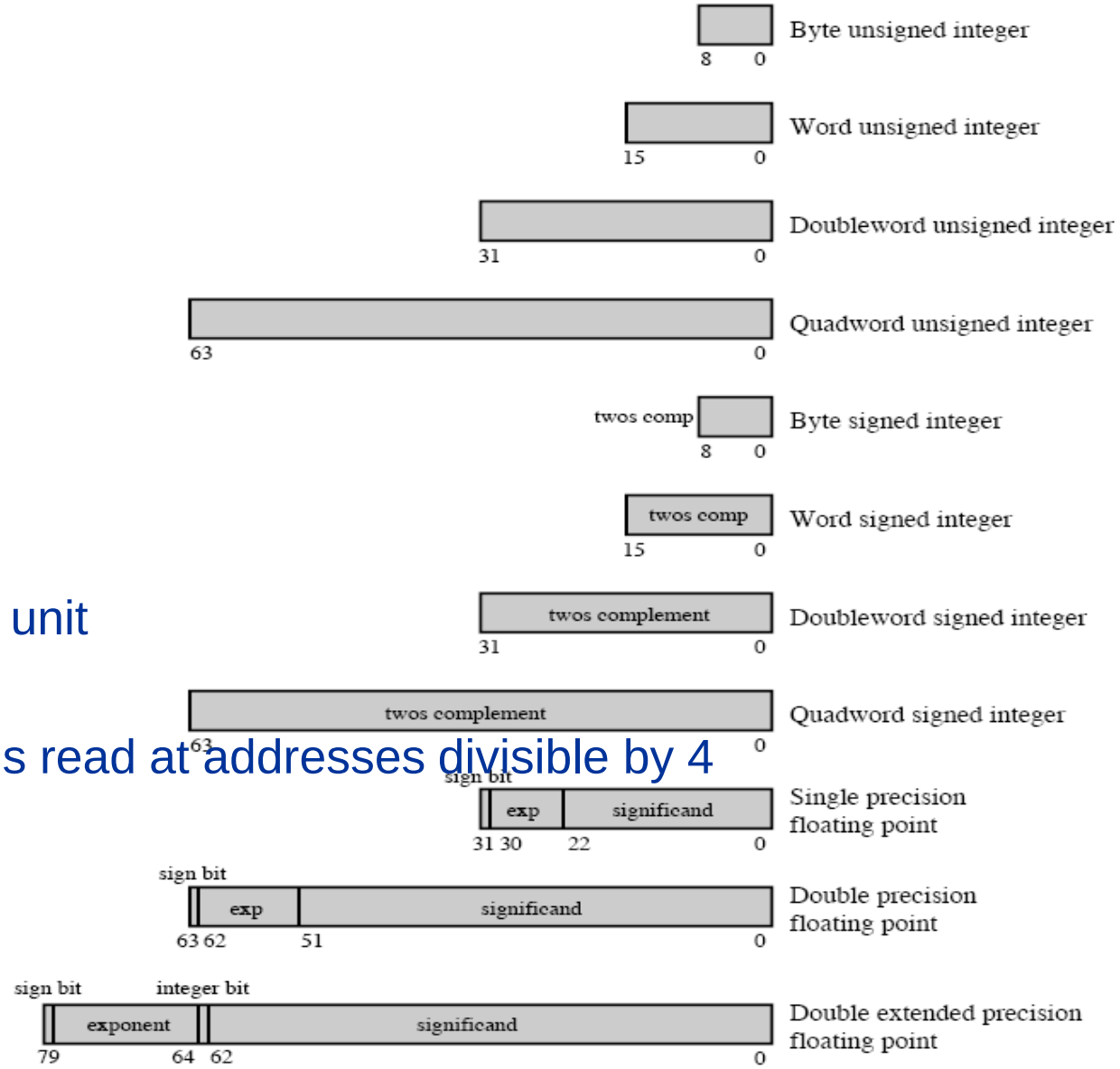
- Xử lý dữ liệu:
  - Thao tác số học: số nguyên, số thực
  - Logique
  - Chuyển đổi
- Chuyển dữ liệu (I/O)
- Lưu dữ liệu (main memory)
- Điều khiển:
  - Kiểm tra và rẽ nhánh
  - Kiểm tra các thanh ghi điều kiện

Phân loại toán hạng

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

Kiểu dữ liệu của Pentium

- 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



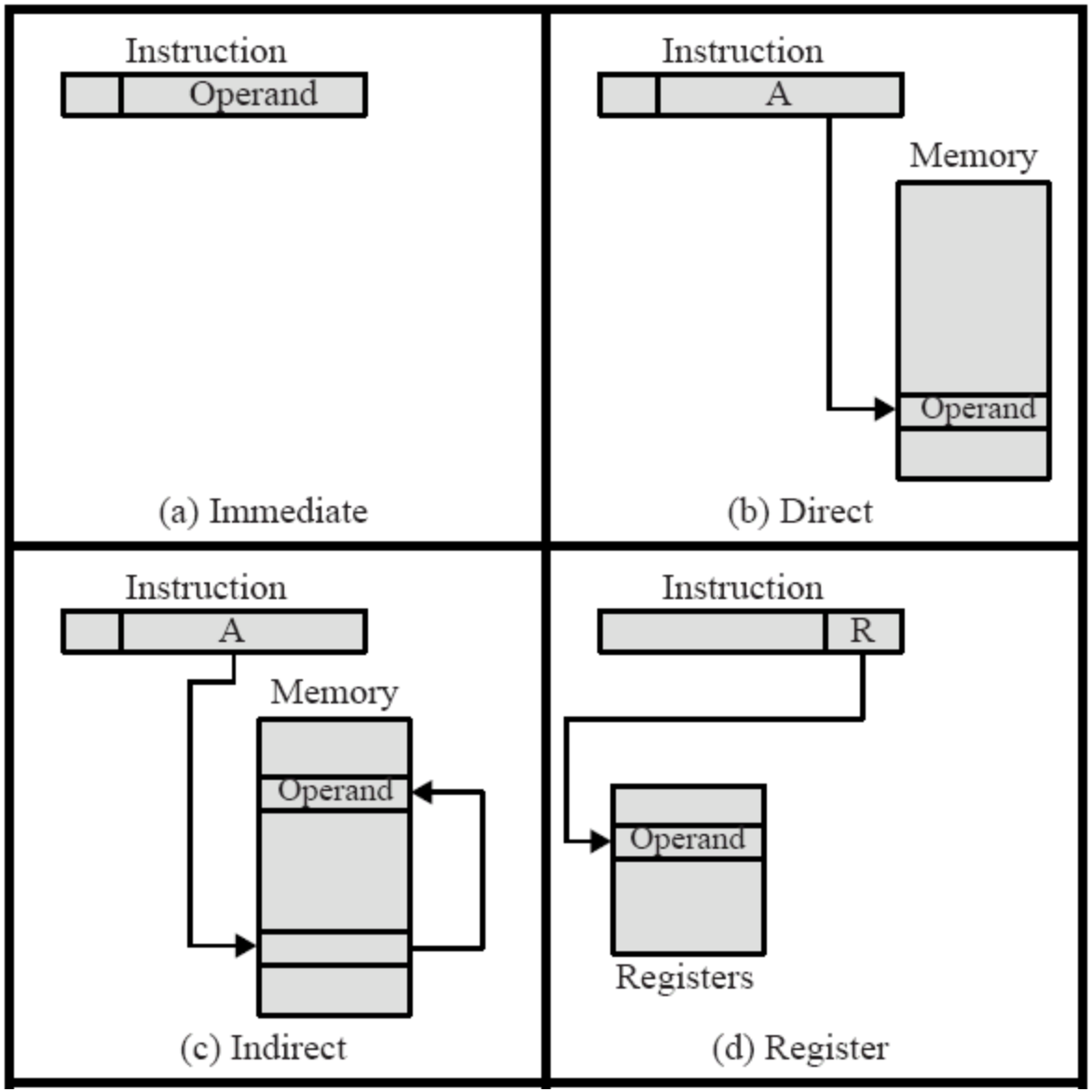
Kiểu dữ liệu của PowerPC

- 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

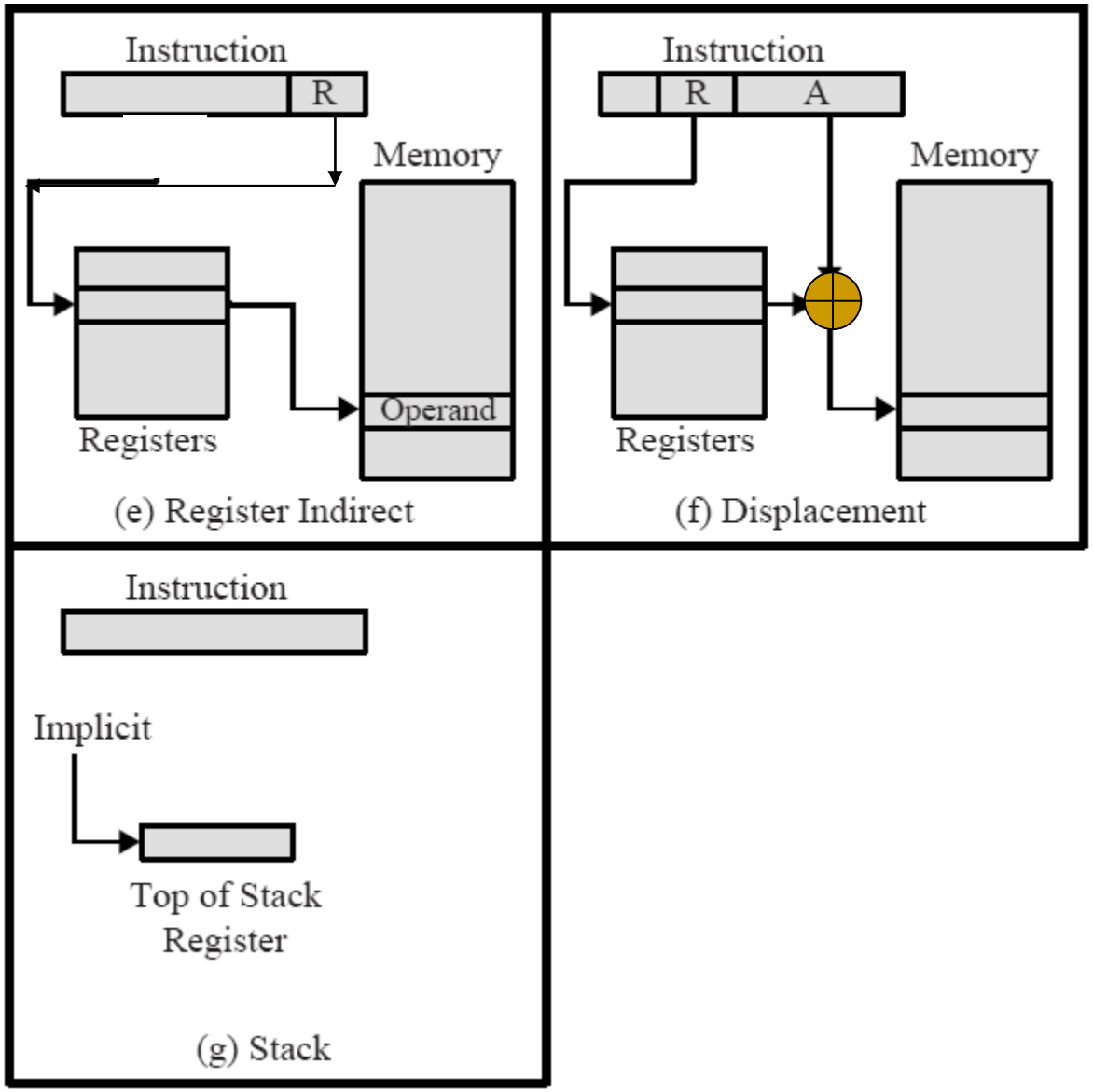
4. Kiểu đánh địa chỉ

- 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



Ví dụ

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

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

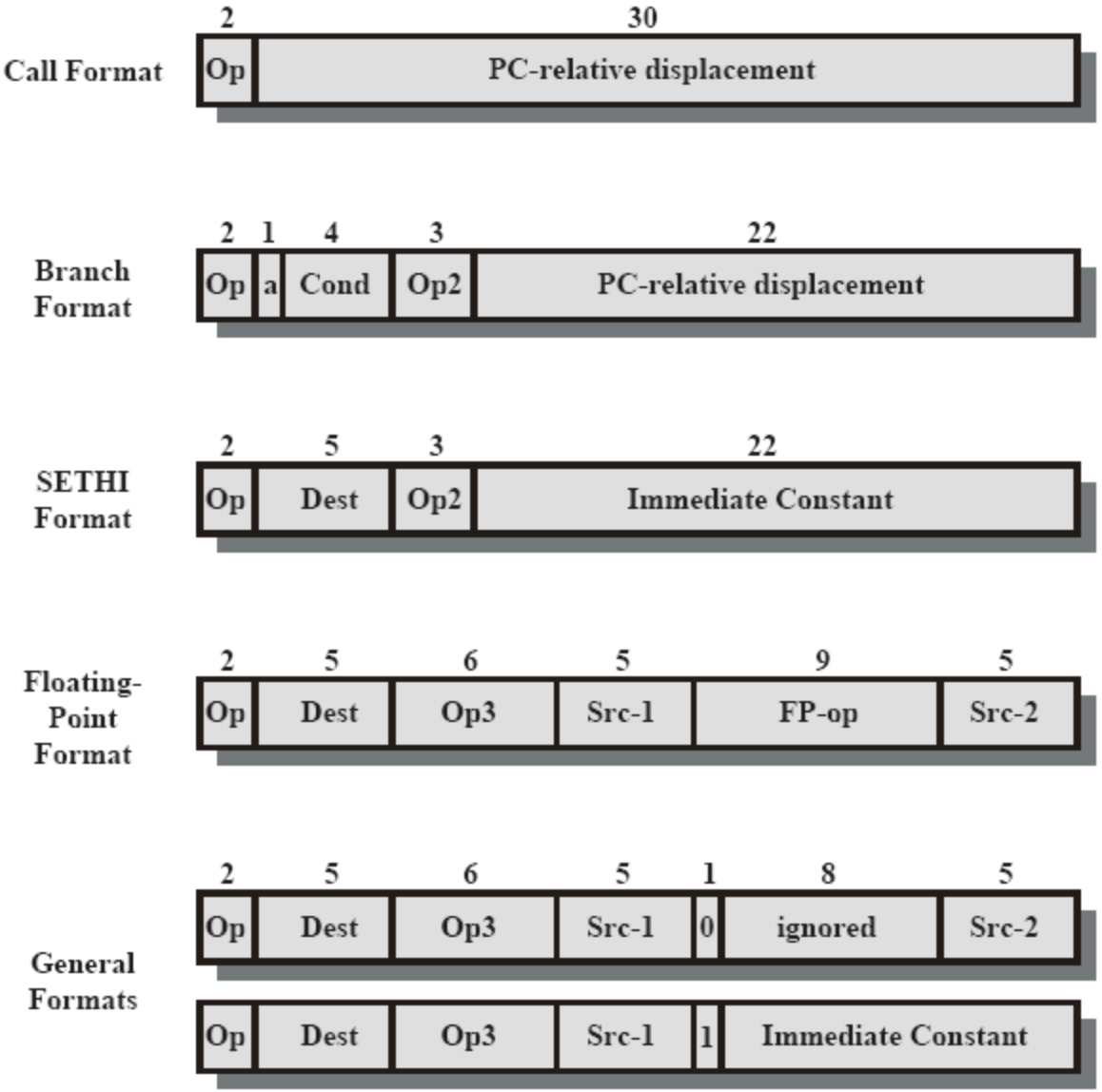
d

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

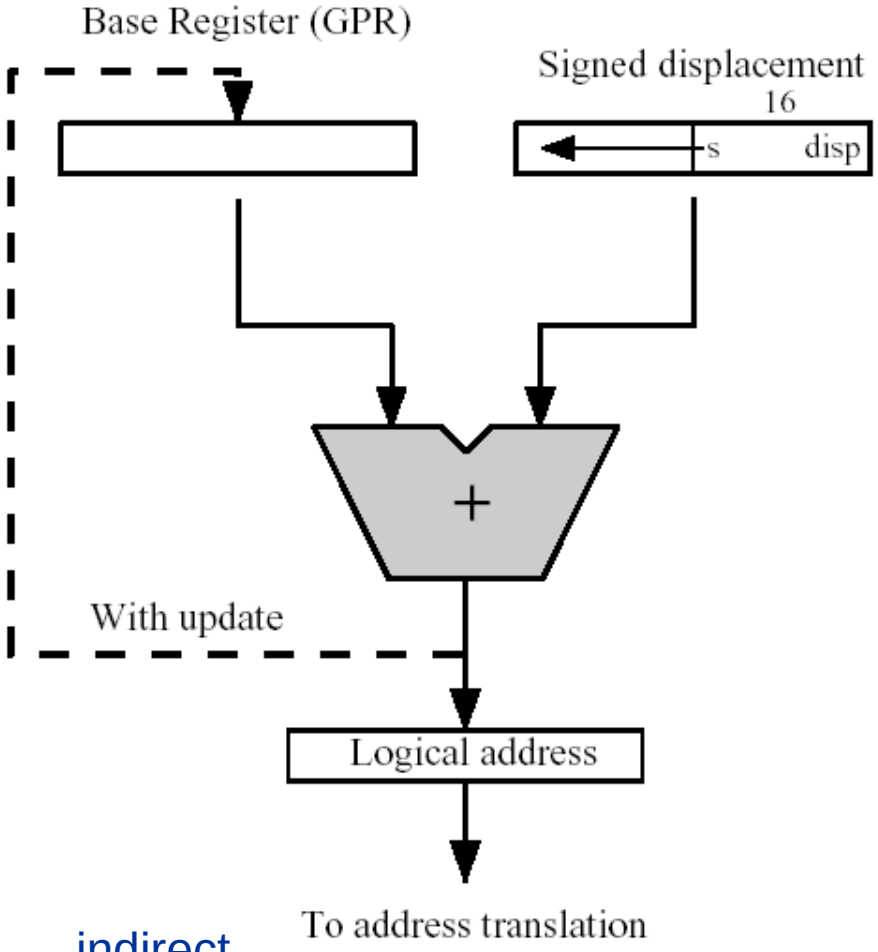
PC = program counter

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

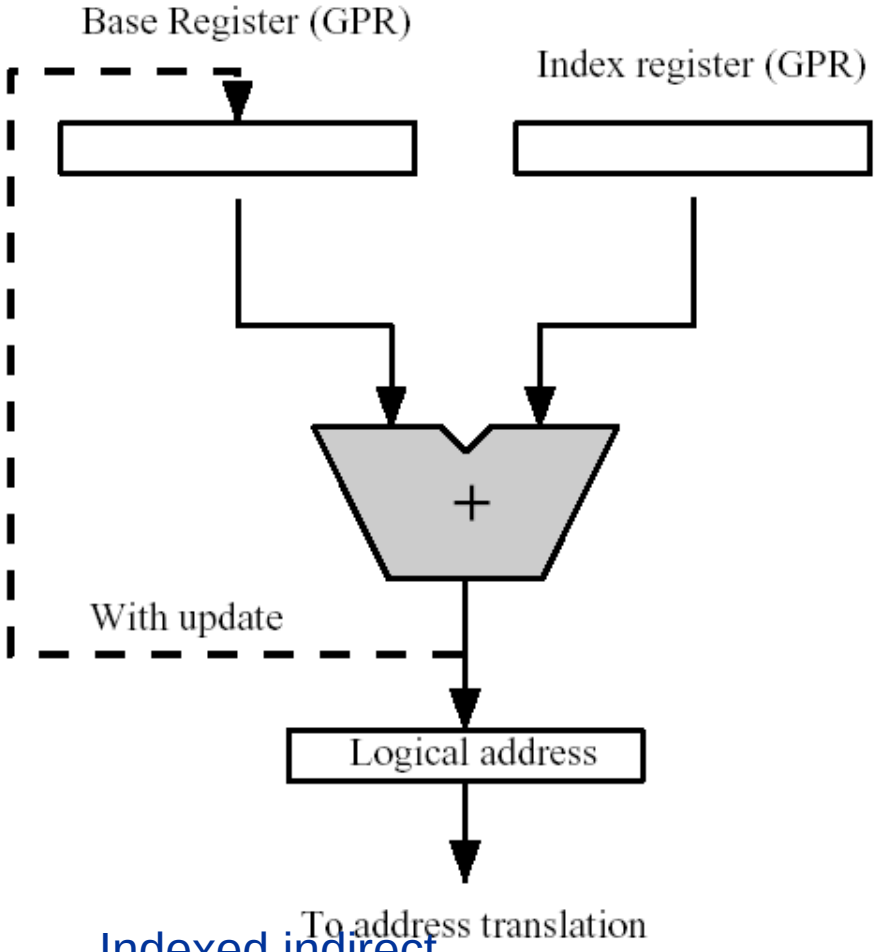
PowerPC...

Các kiểu đánh địa chỉ được phân theo format lệnh :

- Load/Store: indirect và indexed indirect



indirect

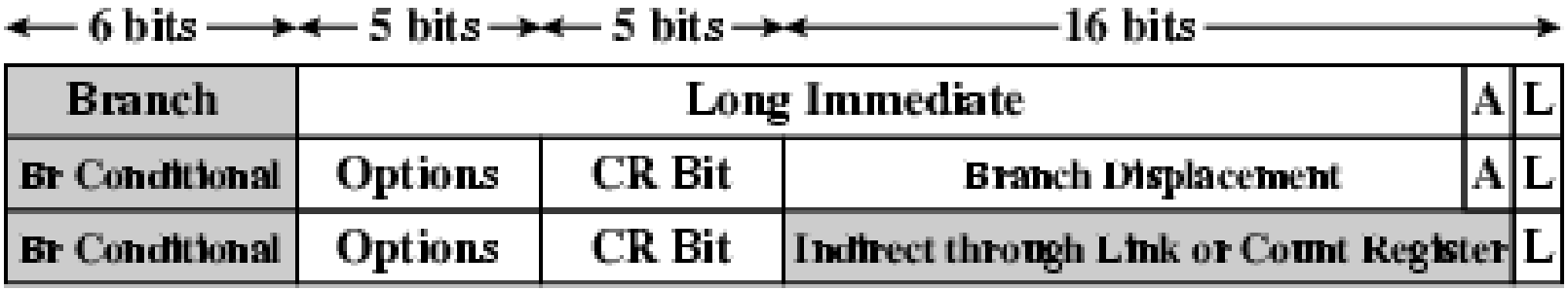


Indexed indirect

PowerPC...

- Rẽ nhánh
  - absolu
  - PC relative
  - indirect
- Tính toán số học
  - register (integer, float)
  - immediate (integer)

PowerPC : format lệnh



(a) Branch instructions

CR	Dest Bit	Source Bit	Source Bit	Add, OR, XOR, etc.	/

(b) Condition register logical instructions

Ld/St Indirect	Dest Register	Base Register	Displacement		
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update	/
Ld/St Indirect	Dest Register	Base Register	Displacement		XO

(c) Load/store instructions

A = Ab
L = Lin
O = Record Overflow in XER
R = Record Conditions in CR1
XO = OpCode Extension
S = Part of Shift Amount Field

PowerPC : format lệnh

Ld/St Indirect	Dest Register	Base Register	Displacement				
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update		/	
Ld/St Indirect	Dest Register	Base Register	Displacement			XO	*

(c) Load/store instructions

Arithmetic	Dest Register	Src Register	Src Register	O	Add, Sub, etc.			R	
Add, Sub, etc.	Dest Register	Src Register	Signed Immediate Value						
Logical	Src Register	Dest Register	Src Register	ADD, OR, XOR, etc.				R	
AND, OR, etc.	Src Register	Dest Register	Unsigned Immediate Value						
Rotate	Src Register	Dest Register	Shift Amt	Mask Begin	Mask End			R	
Rotate or Shift	Src Register	Dest Register	Src Register	Shift Type or Mask				R	
Rotate	Src Register	Dest Register	Shift Amt	Mask		XO	S	R	*
Rotate	Src Register	Dest Register	Src Register	Mask		XO		R	*
Shift	Src Register	Dest Register	Shift Type or Mask				S	R	*

(d) Integer arithmetic, logical, and shift/rotate instructions

Flt sgl/dbl	Dest Register	Src Register	Src Register	Src Register	Fadd, etc.	R

(e) Floating-point arithmetic instructions

A = Absolute or PC relative
L = Link or subroutine
O = Record overflow in XER
R = Record condition in CR1
XO = Opcode extension
S = Part of shift amount field
* = 64-bit implementation only

Pentium - INTEL

- 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

Pentium - INTEL: 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

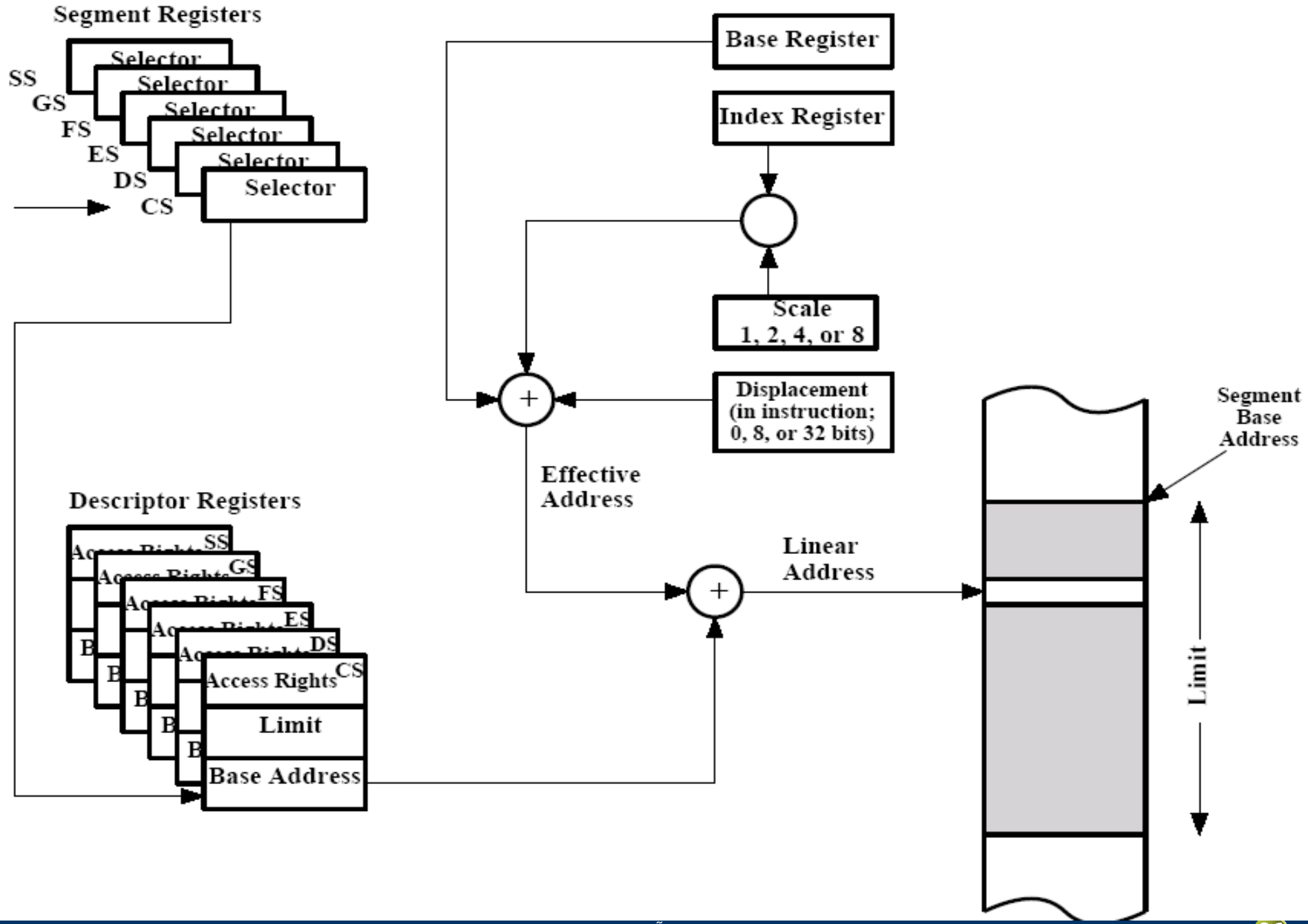
  content of segment register + content of base register + displacement contained in the instruction
- base + index + displacement

  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

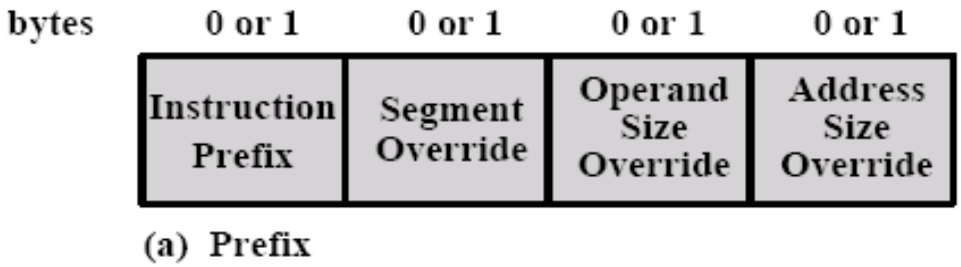
  content of PC + displacement contained in the instruction


Segment Registers
SS
GS
FS
ES
DS
CS
Selector
Selector
Selector
Selector
Selector
Selector
Base Register
Index Register
Scale
1, 2, 4, or 8
Displacement
(in instruction;
0, 8, or 32 bits)
Effective
Address
Linear
Address
Descriptor Registers
Access Rights
Limit
Base Address
Segment
Base
Address
Limit

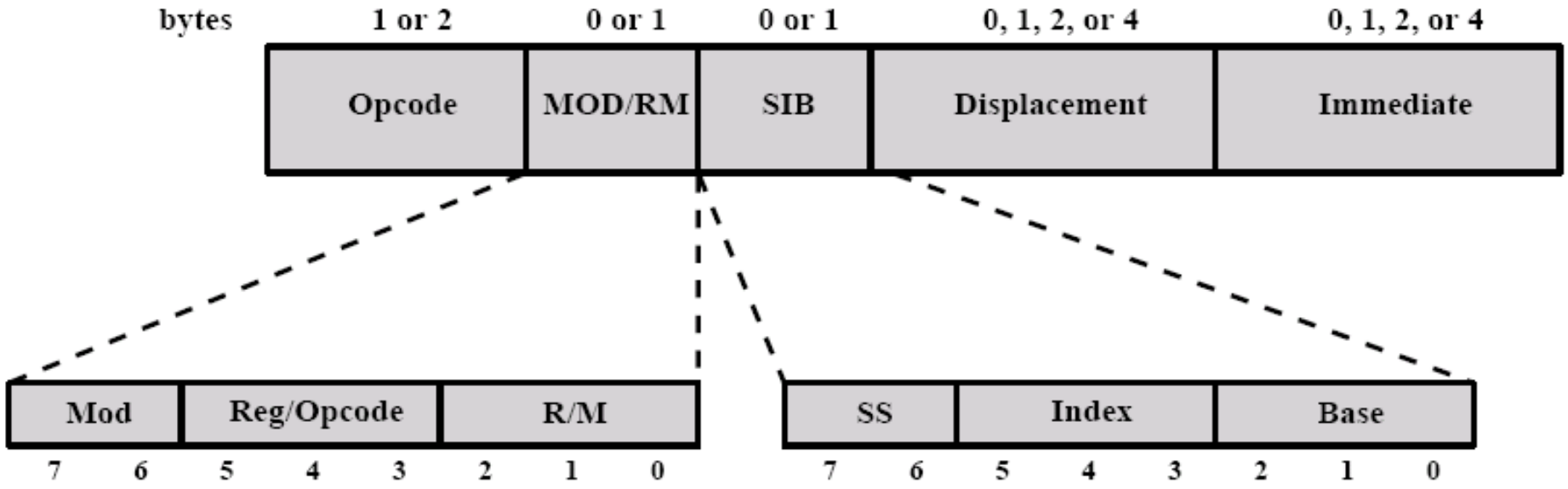
Pentium: format lệnh

- 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

Pentium: format lệnh



(a) Prefix



(b) Instruction

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

Accumulator

Opcode 8 bits	Address 16 bits

```
LOAD    B
ADD     C
STORE   A
ADD     C
STORE   B
SUBINV  A
STORE   D
```

7 × 24 bits = 168 bits

Memory-Memory

Opcode 8 bits	Source 1 16 bits	Source 2 16 bits	Destination 16 bits

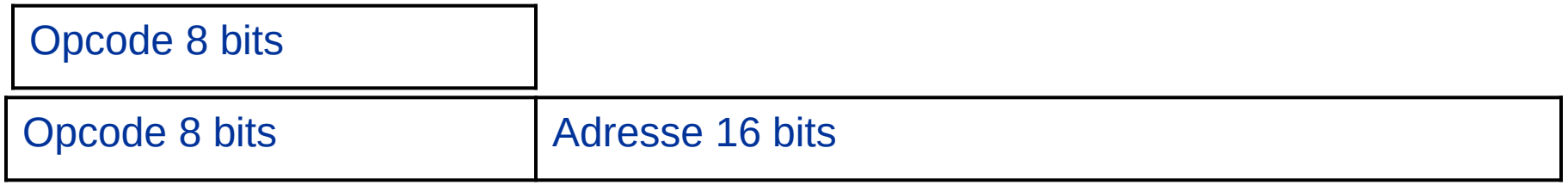
ADD B, C, A

ADD A, C, B

SUB B, A, D

3 × 56 bits = 168 bits

Stack




```
PUSH    B          ADD
PUSH C             POP     B
ADD                PUSH    A
POP A              PUSH    B
PUSH A             SUB
PUSH C             POP D
```

9 × 24 bits + 3 × 8 bits = 240 bits

Load-Store

Opcode 8 bits	Ri 4 bits	xx 4 bits	Adresse 16 bits	
Opcode 8 bits	R1 4 bits	R2 4 bits	Rd 4 bits	xx 4 bits

```
LOAD Rb, B          ADD     Ra, Rc, Rb
LOAD Rc, C          STORE   Rb. B
ADD Rb, Rc, Ra      SUB     Rb, Ra, Rd
STORE Ra, A         STORE   Rd, D
```

5 × 32 bits + 3 × 24 bits = 232 bits

Memory Access

- accumulator

  7 phép chuyển dữ liệu (7 × 32 = 224)
  + độ lớn code (168)
  = 392 bits for 7 instructions (56 bits/instruction)

- memory-memory

  9 phép chuyển dữ liệu (9 × 32 = 288)
  + độ lớn code (168)
  = 456 bits for 3 instructions (152 bits/instructions)

- stack

  9 phép chuyển dữ liệu (9 × 32 = 288)
  + độ lớn code (240)
  = 528 bits for 12 instructions (44 bits/instructions)

- load-store

  5 phép chuyển dữ liệu (5 × 32 = 160)
  + độ lớn code (232)
  = 392 bits for 8 instructions (49 bits/instructions)