Chương 4

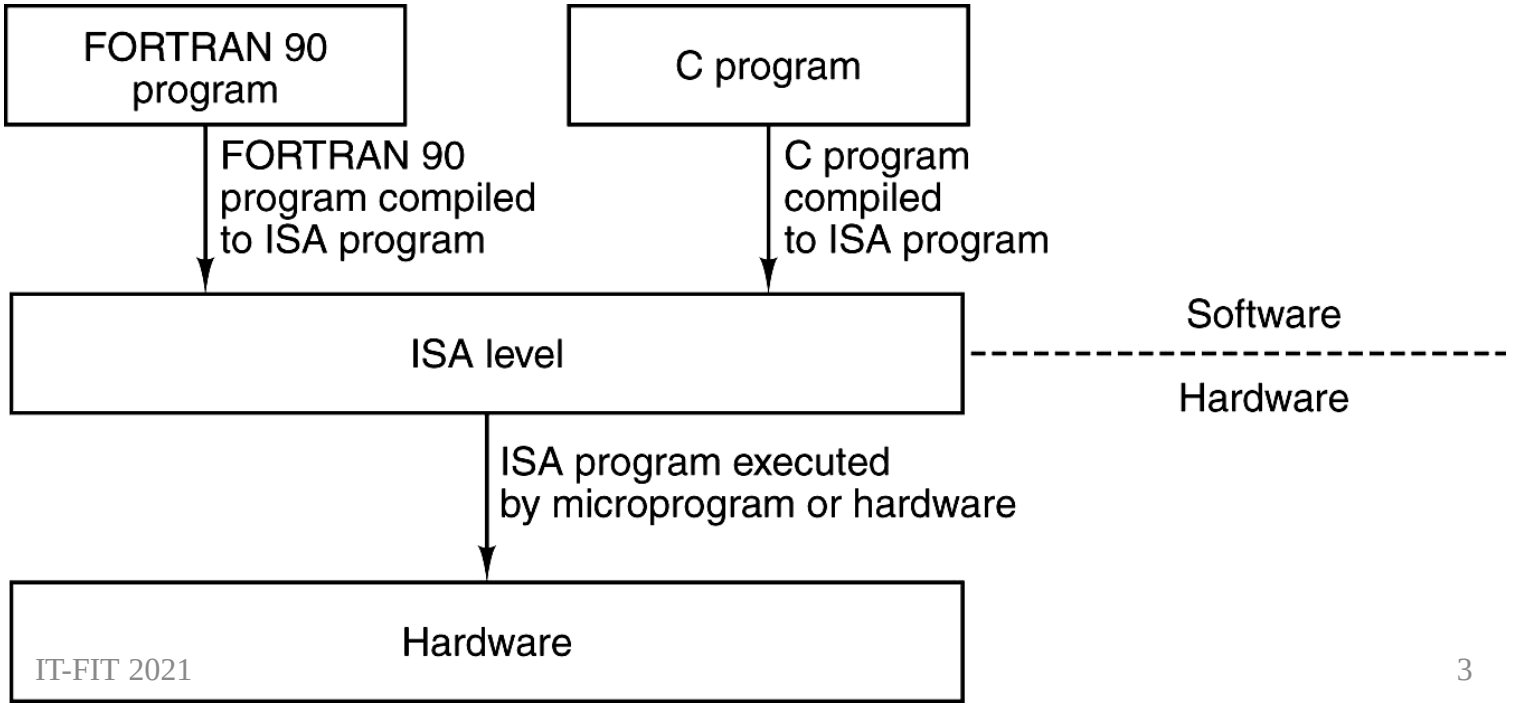
Kiến trúc tập lệnh
(Instruction Set Architecture)

Nội dung

- Mô hình lập trình của máy tính
- Các đặc trưng của lệnh máy
- Các kiểu thao tác của lệnh
- Các phương pháp định địa chỉ
- Phân loại tập lệnh
- Kiến trúc tập lệnh Intel x86

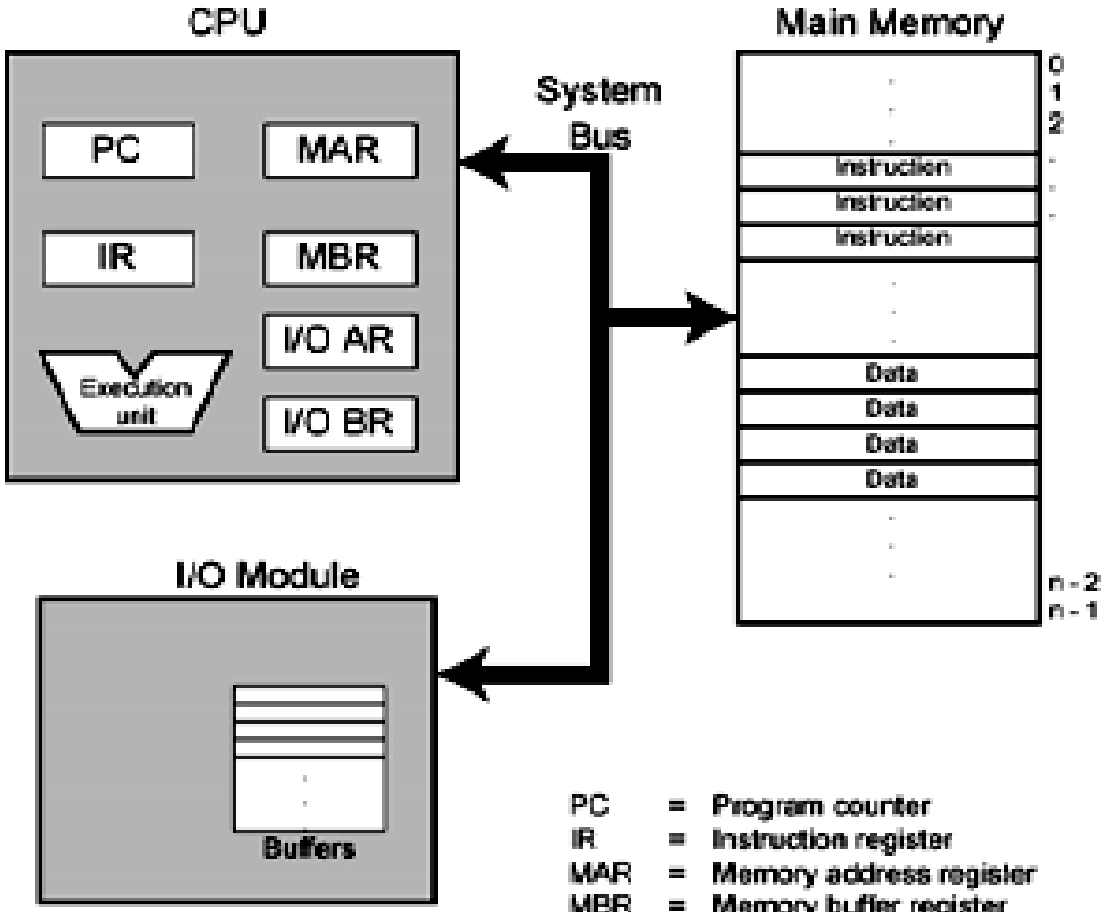
Mô hình lập trình của máy tính

- Vị trí kiến trúc tập lệnh ISA trong máy tính
  - Nằm giữa phần cứng và NNLT cấp cao HLL
  - Giúp phần mềm tương thích khi kiến trúc phần cứng thay đổi



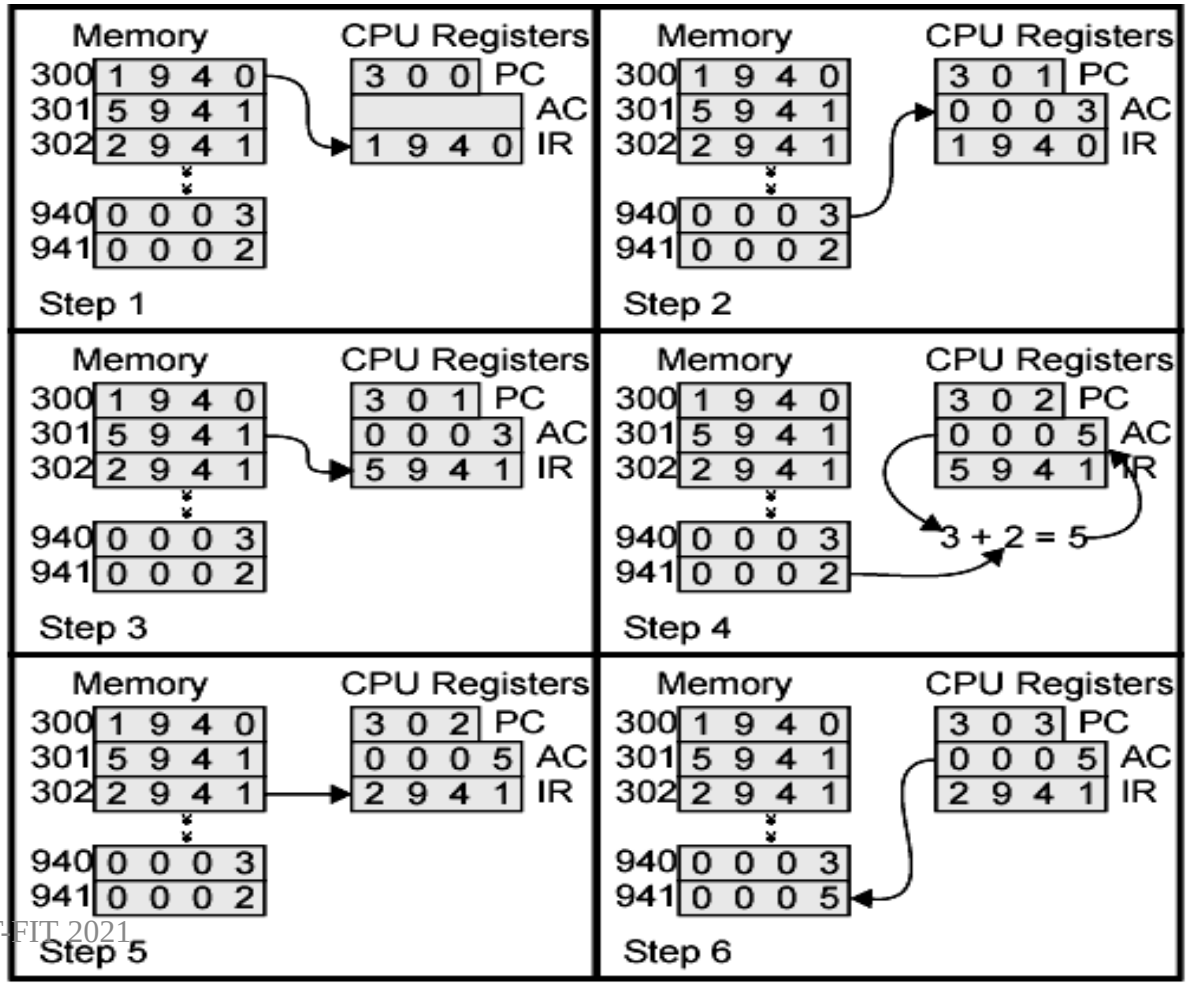
Mô hình lập trình của máy tính

- Máy tính theo quan điểm lập trình



Mô hình lập trình của máy tính

- Ví dụ về sự thi hành chương trình



Mô hình lập trình của máy tính

- Tập thanh ghi (Registers)
  - Chứa các thông tin tạm thời phục vụ cho hoạt động ở thời điểm hiện tại của CPU
  - Được coi là mức đầu tiên của hệ thống bộ nhớ
  - Số lượng thanh ghi nhiều → tăng hiệu năng của CPU
  - Có hai loại thanh ghi:
    - Các thanh ghi lập trình được
    - Các thanh ghi không lập trình được

Mô hình lập trình của máy tính

- Phân loại thanh ghi theo chức năng
  - Thanh ghi địa chỉ: quản lý địa chỉ của bộ nhớ hay cổng IO.
  - Thanh ghi dữ liệu: chứa tạm thời các dữ liệu.
  - Thanh ghi đa năng: có thể chứa địa chỉ hoặc dữ liệu.
  - Thanh ghi điều khiển/trạng thái: chứa các thông tin điều khiển và trạng thái của CPU.
  - Thanh ghi lệnh: chứa lệnh đang được thực hiện.

Mô hình lập trình của máy tính

- Một số thanh ghi điển hình
  - Các thanh ghi địa chỉ (Address Register)
    - Bộ đếm chương trình PC (Program Counter)
    - Con trỏ dữ liệu DP (Data Pointer)
    - Con trỏ ngăn xếp SP (Stack Pointer)
    - Thanh ghi cơ sở và thanh ghi chỉ số (Base Register & Index Register)
  - Các thanh ghi dữ liệu (Data Register)
  - Thanh ghi trạng thái (Status Register)

Mô hình lập trình của máy tính

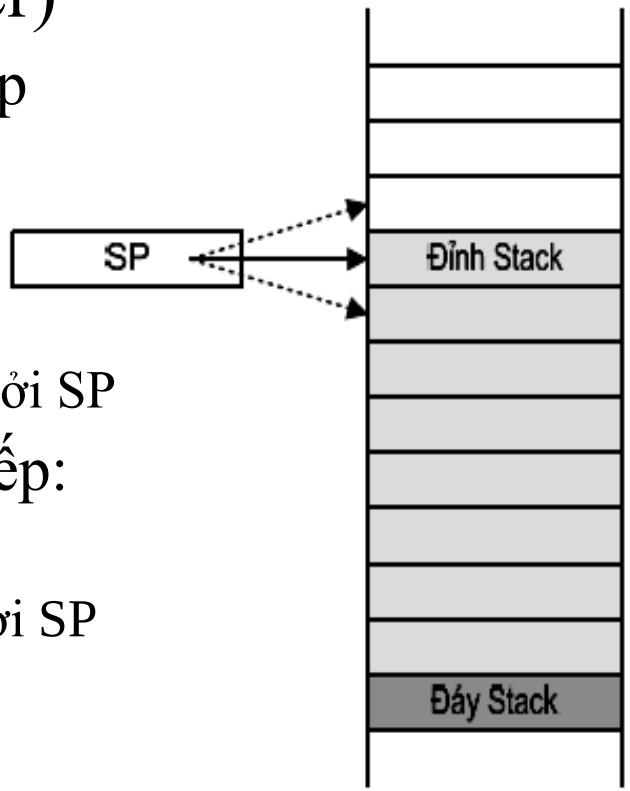
- Bộ đếm chương trình PC
  - Còn được gọi là con trỏ lệnh IP (Instruction Pointer)
  - Giữ địa chỉ của lệnh tiếp theo sẽ được thi hành.
  - Sau khi một lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp.
- Thanh ghi con trỏ dữ liệu DP
  - Chứa địa chỉ của ô nhớ dữ liệu mà CPU muốn truy cập
  - Thường có nhiều thanh ghi con trỏ dữ liệu cho phép chương trình có thể truy cập nhiều vùng nhớ đồng thời.

Mô hình lập trình của máy tính

- Ngăn xếp (Stack)
  - Ngăn xếp là vùng nhớ có cấu trúc LIFO (Last In - First Out) hoặc FILO (First In - Last Out)
  - Ngăn xếp thường dùng để phục vụ cho chương trình con
  - Đáy ngăn xếp là một ô nhớ xác định
  - Đỉnh ngăn xếp là thông tin nằm ở vị trí trên cùng trong ngăn xếp
  - Đỉnh ngăn xếp có thể bị thay đổi

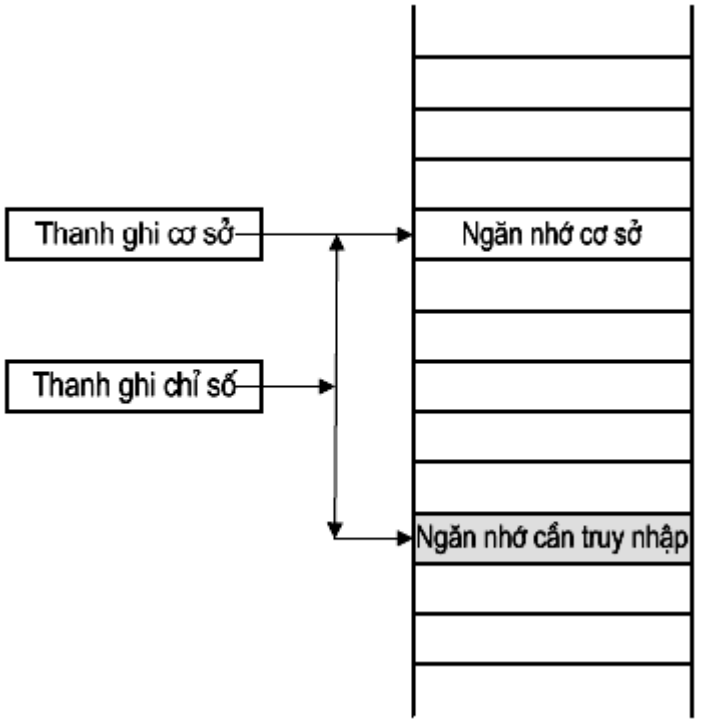
Mô hình lập trình của máy tính

- Con trỏ ngăn xếp SP (Stack Pointer)
  - Chứa địa chỉ của ô nhớ đỉnh ngăn xếp
  - Khi cất một thông tin vào ngăn xếp:
    - Thao tác PUSH
    - Nội dung của SP tự động tăng
    - Thông tin được cất vào ô nhớ đang trỏ bởi SP
  - Khi lấy một thông tin ra khỏi ngăn xếp:
    - Thao tác POP
    - Thông tin được đọc từ ô nhớ đang trỏ bởi SP
    - Nội dung của SP tự động giảm
  - Khi ngăn xếp rỗng, SP trỏ vào đáy



Mô hình lập trình của máy tính

- Thanh ghi cơ sở và thanh ghi chỉ số
  - Thanh ghi cơ sở: chứa địa chỉ của ngăn nhớ cơ sở (địa chỉ cơ sở)
  - Thanh ghi chỉ số: chứa độ lệch địa chỉ giữa ngăn nhớ mà CPU cần truy cập so với ngăn nhớ cơ sở (chỉ số)
  - Địa chỉ của ngăn nhớ cần truy cập = địa chỉ cơ sở + chỉ số



Mô hình lập trình của máy tính

- Thanh ghi dữ liệu (Data Register)
  - Chứa các dữ liệu tạm thời hoặc các kết quả trung gian
  - Cần có nhiều thanh ghi dữ liệu
  - Các thanh ghi số nguyên: 8, 16, 32, 64 bit
  - Các thanh ghi số dấu chấm động: 32, 64, 80 bit
- Thanh ghi trạng thái (Status Register)
  - Còn gọi là thanh ghi cờ (Flags Register) hoặc từ trạng thái chương trình PSW (Program Status Word)
  - Chứa các thông tin trạng thái của CPU
    - Các cờ phép toán: báo hiệu trạng thái của kết quả phép toán
    - Các cờ điều khiển: biểu thị trạng thái điều khiển của CPU

Mô hình lập trình của máy tính

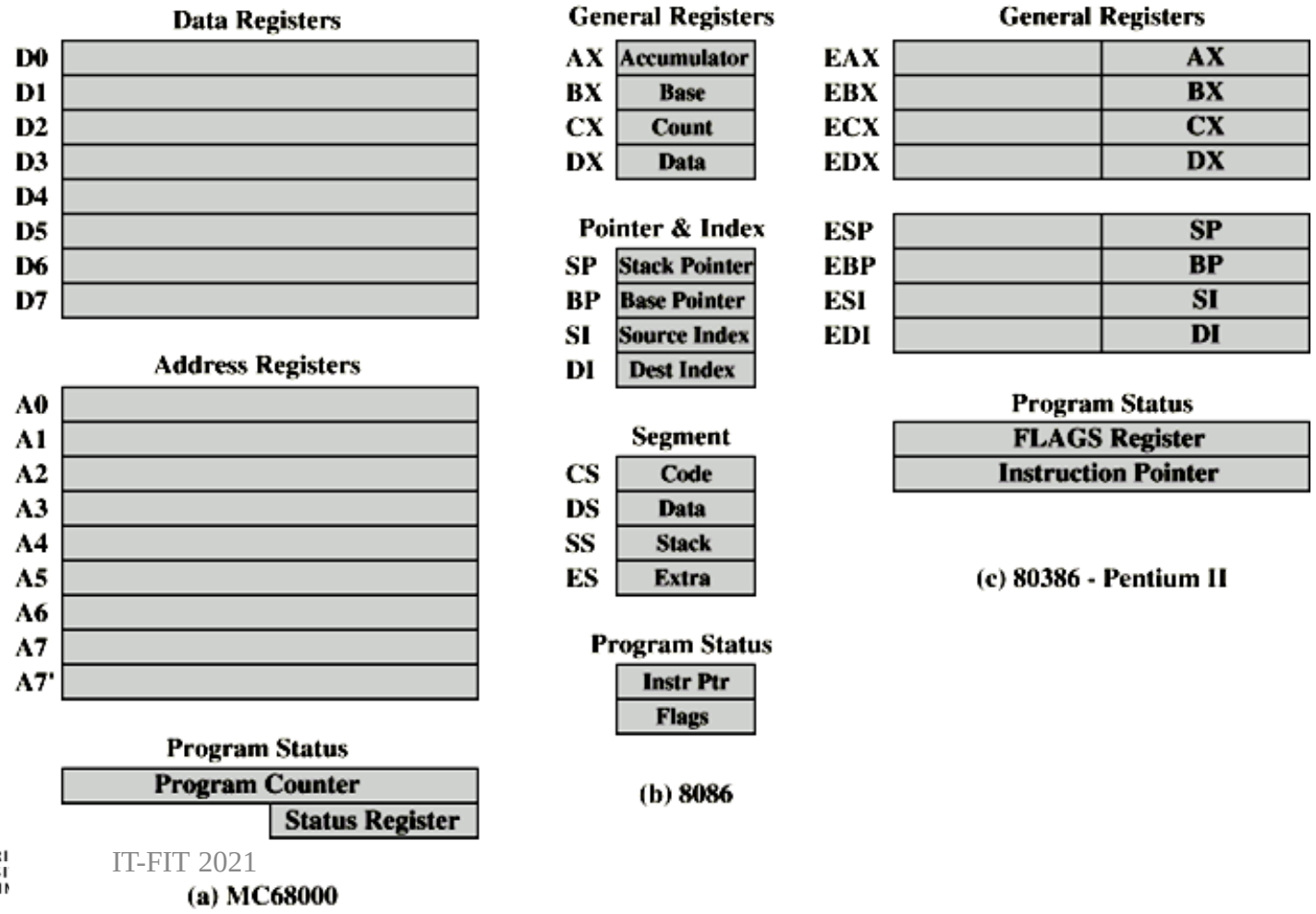
- Ví dụ cờ phép toán
  - Zero Flag (cờ rỗng): được thiết lập lên 1 khi kết quả của phép toán bằng 0.
  - Sign Flag (cờ dấu): được thiết lập lên 1 khi kết quả phép toán nhỏ hơn 0 (kết qủa âm)
  - Carry Flag (cờ nhớ): được thiết lập lên 1 nếu phép toán có nhớ ra ngoài bit cao nhất → cờ báo tràn với số không dấu.
  - Overflow Flag (cờ tràn): được thiết lập lên 1 nếu cộng hai số nguyên cùng dấu mà kết quả có dấu ngược lại → cờ báo tràn với số có dấu .

Mô hình lập trình của máy tính

- Ví dụ cờ điều khiển
  - Interrupt Flag (Cờ cho phép ngắt):
    - Nếu IF = 1 → CPU ở trạng thái cho phép ngắt với tín hiệu yêu cầu ngắt từ bên ngoài gửi tới
    - Nếu IF = 0 → CPU ở trạng thái cấm ngắt với tín hiệu yêu cầu ngắt từ bên ngoài gửi tới
  - Direction Flag (Cờ hướng):
    - Nếu DF=0 → Truy cập bộ nhớ theo hướng tăng
    - Nếu DF=1 → Truy cập bộ nhớ theo hướng giảm

Mô hình lập trình của máy tính

- Ví dụ: Tập thanh ghi của một số bộ xử lý



Các đặc trưng của lệnh máy

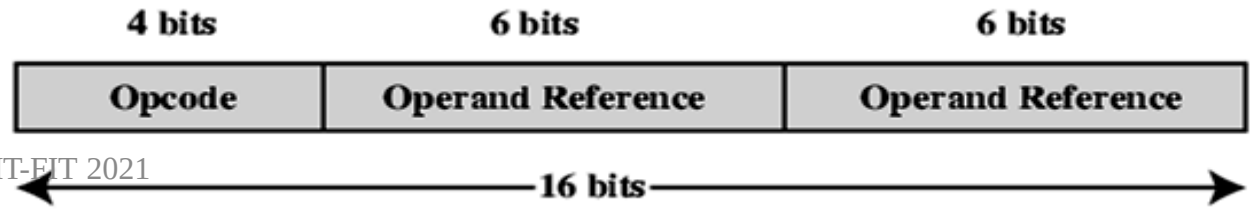
- Giới thiệu chung về tập lệnh
  - Mỗi bộ xử lý có một tập lệnh xác định
  - Tập lệnh thường có hàng chục đến hàng trăm lệnh
  - Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.
  - Các lệnh được mô tả bằng các ký hiệu gợi nhớ → chính là các lệnh của hợp ngữ (assembly), ví dụ: ADD, SUB, LOAD, STORE,…

Các đặc trưng của lệnh máy

- Các thành phần của lệnh máy

Opcode	Operand address

  - Mã thao tác (operation code): mã hóa cho thao tác mà bộ xử lý phải thực hiện bằng số nhị phân (làm gì?)
  - Địa chỉ toán hạng (operand address): chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động (làm ở đâu?)
    - Toán hạng nguồn: dữ liệu vào của thao tác
    - Toán hạng đích: dữ liệu ra của thao tác
    - Toán hạng: Thanh ghi, bộ nhớ, thiết bị ngoại vi,…
    - Ví dụ: 1 lệnh 16 bit có 2 toán hạng



Các đặc trưng của lệnh máy

- Số lượng địa chỉ toán hạng trong lệnh
  - Ba địa chỉ toán hạng:
    - 2 toán hạng nguồn, 1 toán hạng đích
    - Ví dụ : a = b + c → ADD A, B, C
    - Từ lệnh dài vì phải mã hoá địa chỉ cho cả ba toán hạng
  - Hai địa chỉ toán hạng:
    - Một toán hạng vừa là toán hạng nguồn vừa là toán hạng đích; toán hạng còn lại là toán hạng nguồn
    - Ví dụ : a = a + b → ADD A, B
    - Giá trị cũ của 1 toán hạng nguồn bị mất vì phải chứa kết quả
    - Rút gọn độ dài từ lệnh, được sử dụng phổ biến

Các đặc trưng của lệnh máy

- Số lượng địa chỉ toán hạng trong lệnh (tiếp)
  - Một địa chỉ toán hạng:
    - Một toán hạng được chỉ ra trong lệnh
    - Một toán hạng là ngầm định, thường là thanh ghi tích lũy (accumulator)
    - Ví dụ : a = b + c

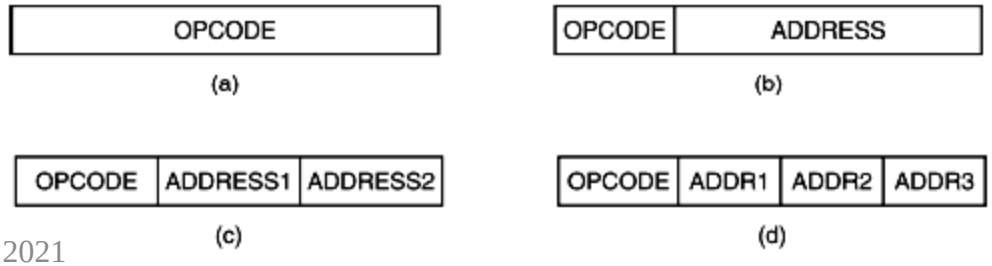
```
LOAD  B
ADD   C
STORE  A
```

  - Không địa chỉ toán hạng:
    - Các toán hạng đều được ngầm định: Sử dụng Stack
    - Ví dụ: a = b + c

```
PUSH   B
PUSH   C
ADD
POP   A
```

Các đặc trưng của lệnh máy

- Đánh giá về số địa chỉ toán hạng
  - Nhiều địa chỉ toán hạng
    - Các lệnh phức tạp hơn
    - Cần nhiều thanh ghi
    - Chương trình có ít lệnh hơn
    - Nhận lệnh và thực hiện lệnh chậm hơn
  - Ít địa chỉ toán hạng
    - Các lệnh đơn giản hơn
    - Cần ít thanh ghi
    - Chương trình có nhiều lệnh hơn
    - Nhận lệnh và thực hiện lệnh nhanh hơn



Các đặc trưng của lệnh máy

- Các kiểu toán hạng
  - Địa chỉ
  - Số
    - Số nguyên
    - Số dấu chấm động
    - Mã BCD
  - Ký tự
    - Mã ASCII
  - Dữ liệu logic
    - Các bit hoặc các cờ

Câu hỏi: Khi đọc trong 1 ô nhớ nhận được gía trị nhị phân 65, làm sao biết được đây là gì?

- Số nguyên 65
- Ký tự 'A'
- Lệnh CT 65
- Địa chỉ 65

Các kiểu thao tác của lệnh

- Phân loại lệnh:
  - Di chuyển dữ liệu
  - Xử lý số học với số nguyên
  - Xử lý logic
  - Điều khiển vào-ra (IO)
  - Chuyển điều khiển (rẽ nhánh)
  - Điều khiển hệ thống

Các kiểu thao tác của lệnh

- Các lệnh di chuyển 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 Hoán 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

Các kiểu thao tác của lệnh

- Các lệnh 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 0 trừ toán hạng)
  - INCREMENT Tăng toán hạng thêm 1
  - DECREMENT Giảm toán hạng đi 1
  - COMPARE Trừ hai toán hạng để lập cờ

Các kiểu thao tác của lệnh

- Các lệnh logic
  - AND Thực hiện phép AND hai toán hạng
  - OR Thực hiện phép OR hai toán hạng
  - XOR Thực hiện phép XOR hai toán hạng
  - NOT Đảo bit của toán hạng (lấy bù 1)
  - TEST Thực hiện phép AND hai toán hạng để lập cờ

Các kiểu thao tác của lệnh

- Ví dụ các lệnh logic
  - Giả sử có hai thanh ghi chứa dữ liệu như sau:

    (R1) = 1010 1010

    (R2) = 0000 1111
  - R1 ← (R1) AND (R2) = 0000 1010

    Phép toán AND dùng để xoá (Clear) một số bit và giữ nguyên một số bit còn lại của toán hạng.
  - R1 ← (R1) OR (R2) = 1010 1111

    Phép toán OR dùng để thiết lập (Set) một số bit và giữ nguyên một số bit còn lại của toán hạng.
  - R1 ← (R1) XOR (R2) = 1010 0101

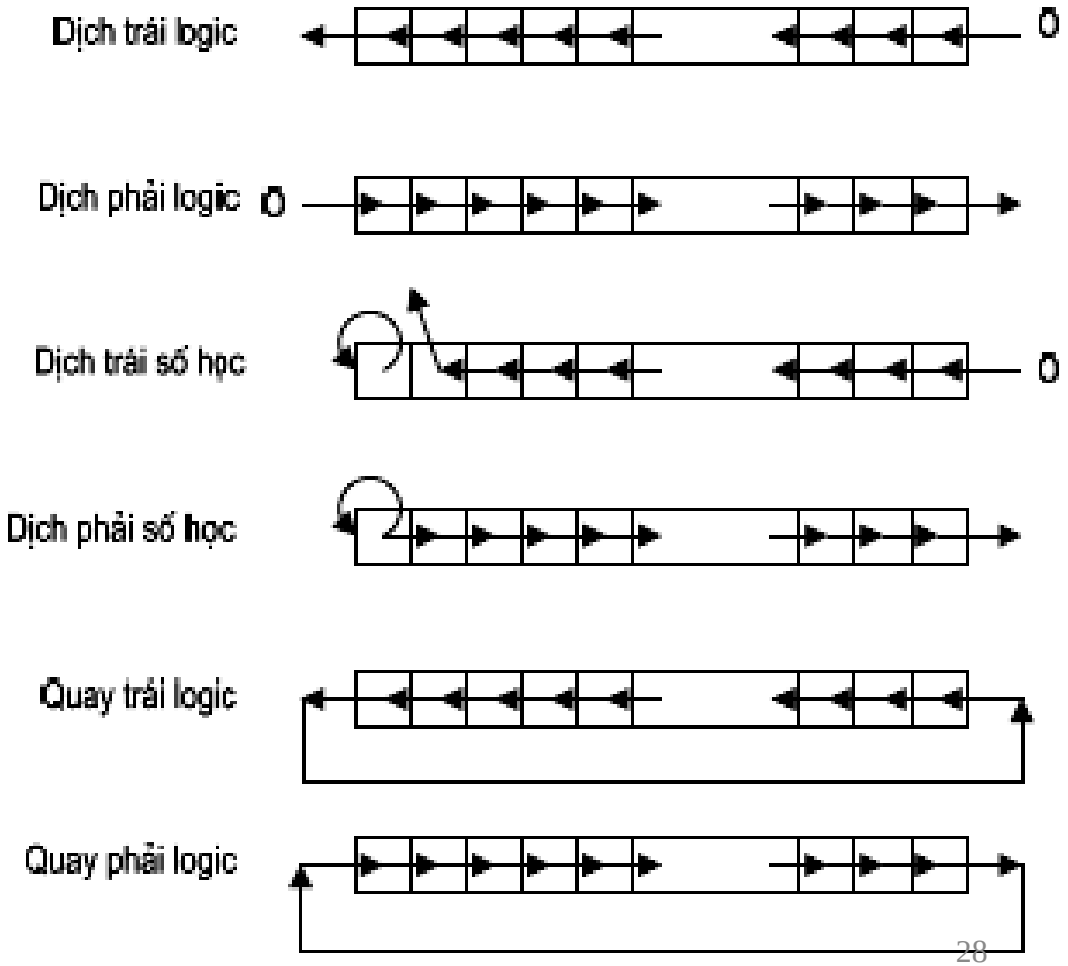
    Phép toán XOR dùng để đảo một số bit và giữ nguyên một số bit còn lại của toán hạng.

Các kiểu thao tác của lệnh

- Các lệnh logic (tiếp)
  - SHIFT

    Dịch trái (phải) toán hạng
  - ROTATE

    Quay trái (phải) toán hạng



Các kiểu thao tác của lệnh

- Các lệnh nhập xuất chuyên dụng
  - INPUT : Copy dữ liệu từ một cổng xác định đưa đến đích (thiết bị → bộ nhớ)
  - OUTPUT: Copy dữ liệu từ nguồn đến một cổng xác định (bộ nhớ → thiết bị)
- Các lệnh chuyển điều khiển
  - JUMP (BRANCH): Lệnh rẽ nhánh không điều kiện
  - CONDITIONAL JUMP : Lệnh rẽ nhánh có điều kiện
  - CALL : Lệnh gọi chương trình con
  - RETURN : Lệnh trở về từ chương trình con

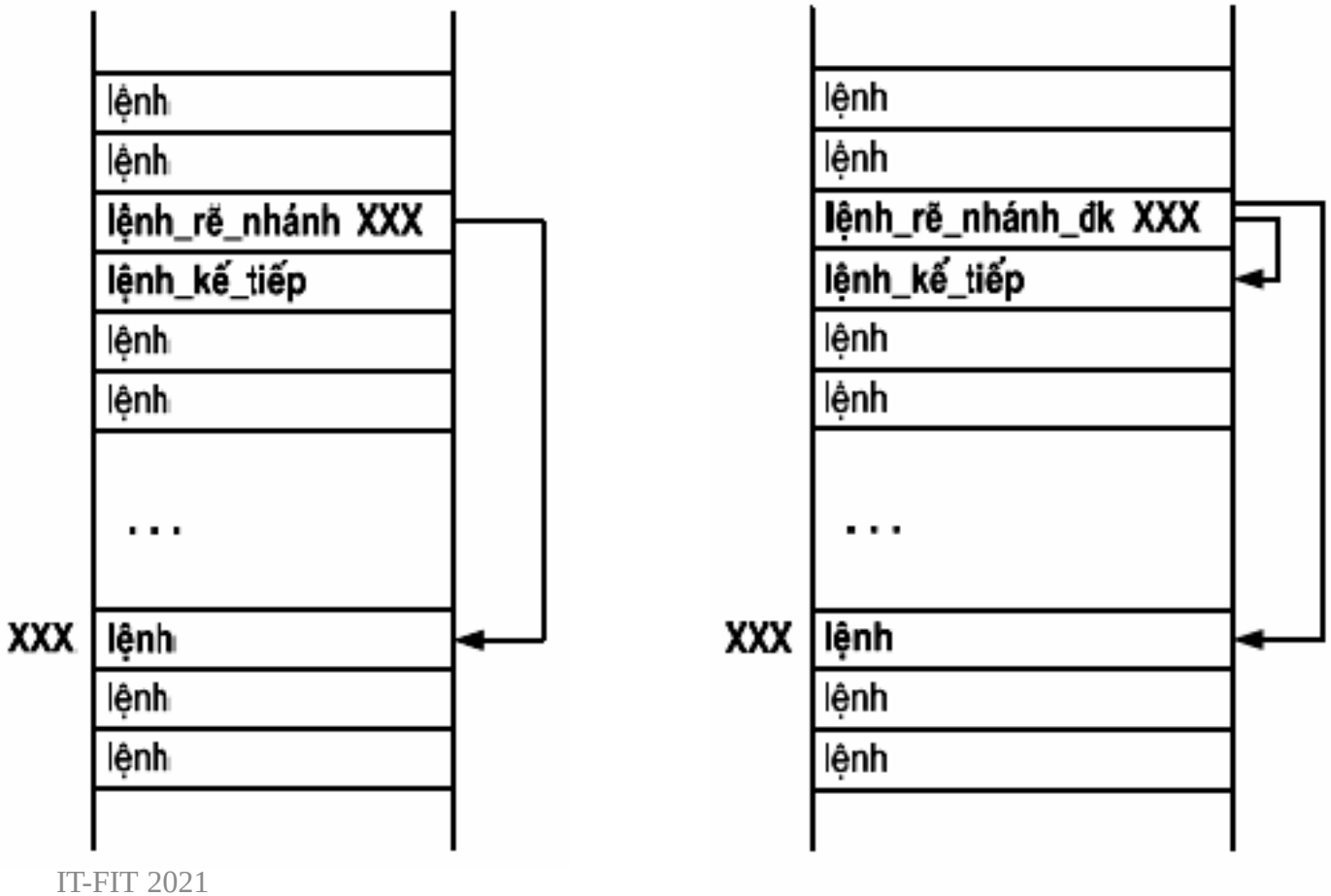
Các kiểu thao tác của lệnh

- Lệnh rẽ nhánh có điều kiện
  - Trong lệnh có kèm theo điều kiện
  - Kiểm tra điều kiện trong lệnh:
    - Nếu điều kiện đúng → chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX

      PC ← XXX
    - Nếu điều kiện sai → chuyển sang thực hiện **lệnh_kế_tiếp**
  - Điều kiện thường được kiểm tra thông qua các cờ
  - Có nhiều lệnh rẽ nhánh theo các điều kiện khác nhau

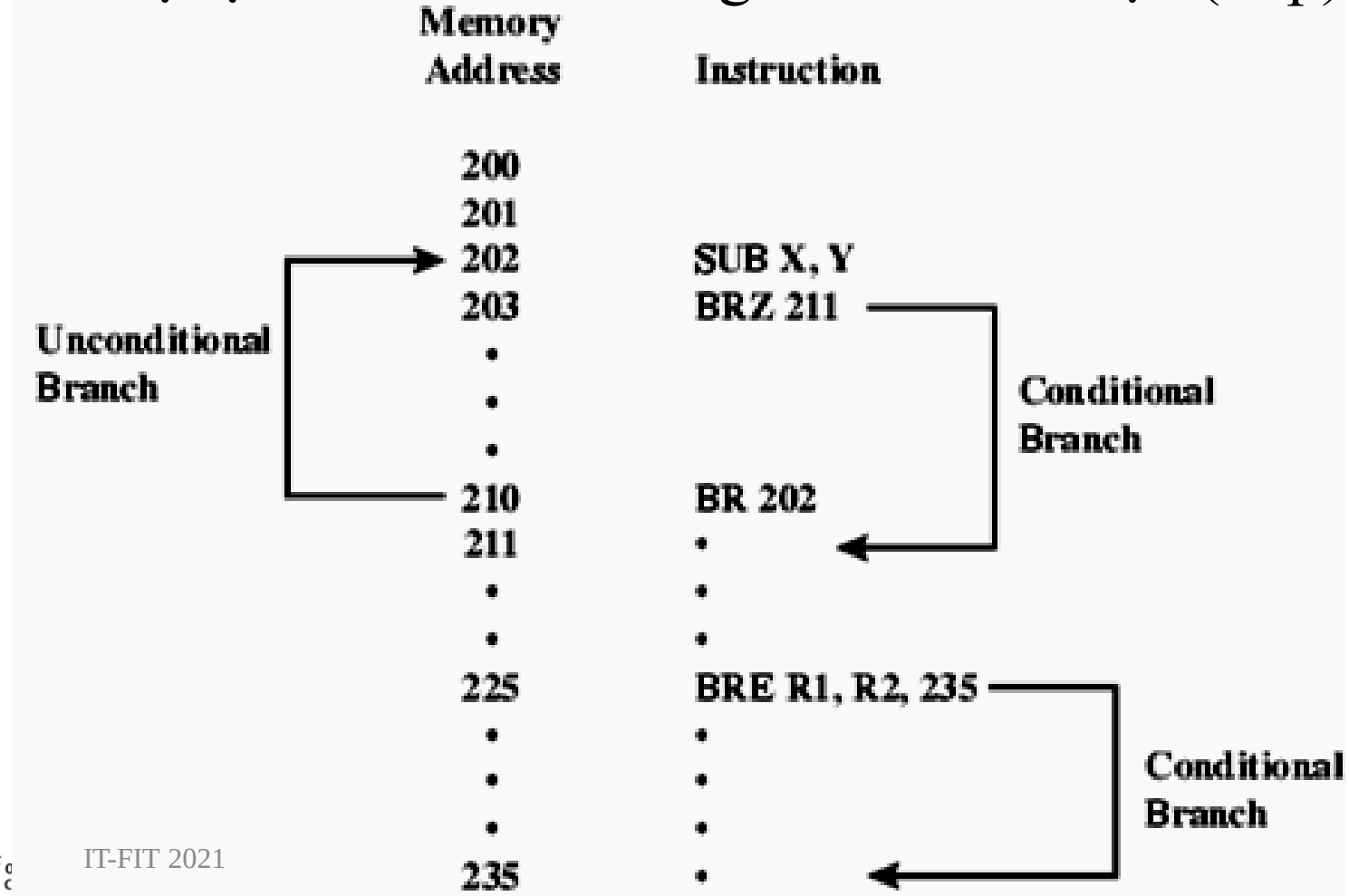
Các kiểu thao tác của lệnh

- Minh hoạ lệnh rẽ nhánh không và có điều kiện



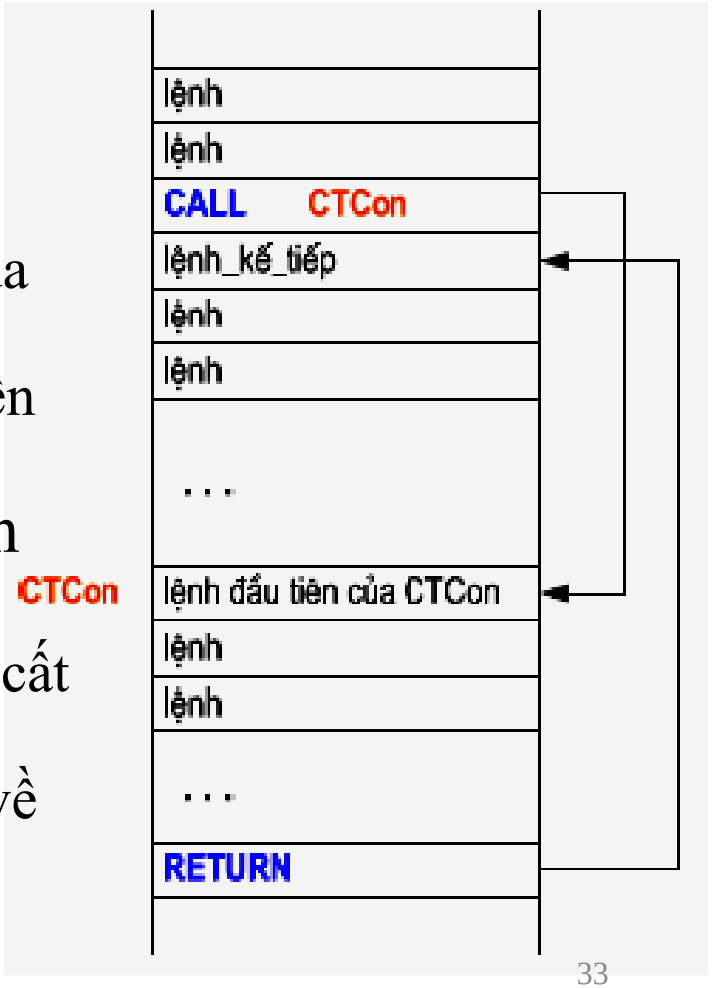
Các kiểu thao tác của lệnh

- Minh hoạ lệnh rẽ nhánh không và có điều kiện (tiếp)



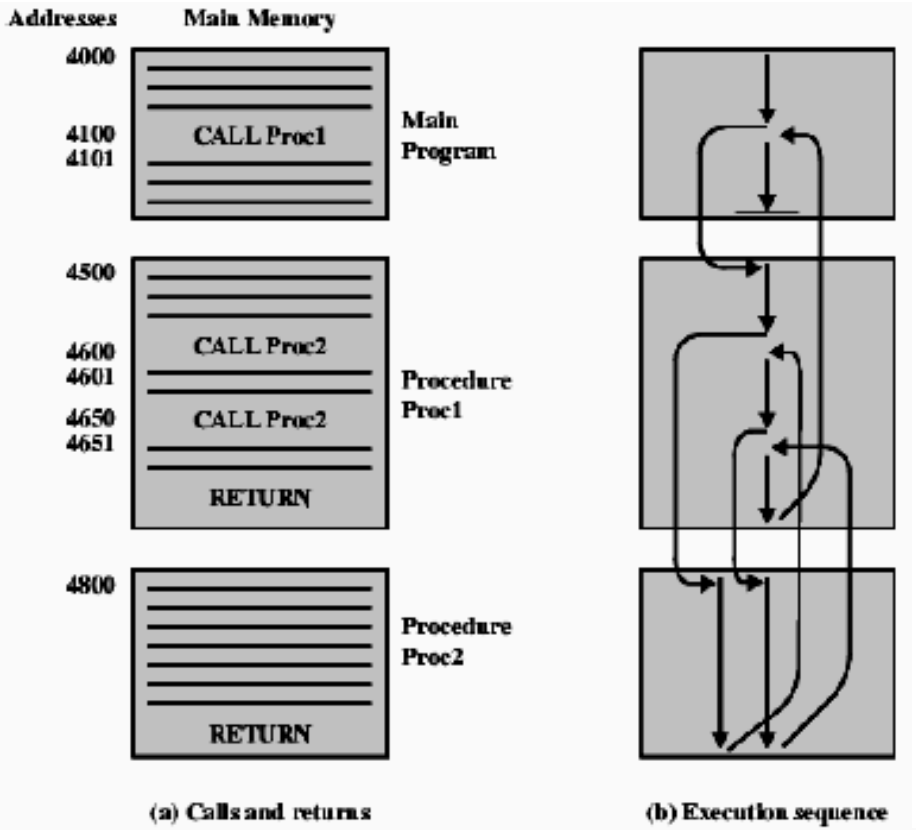
Các kiểu thao tác của lệnh

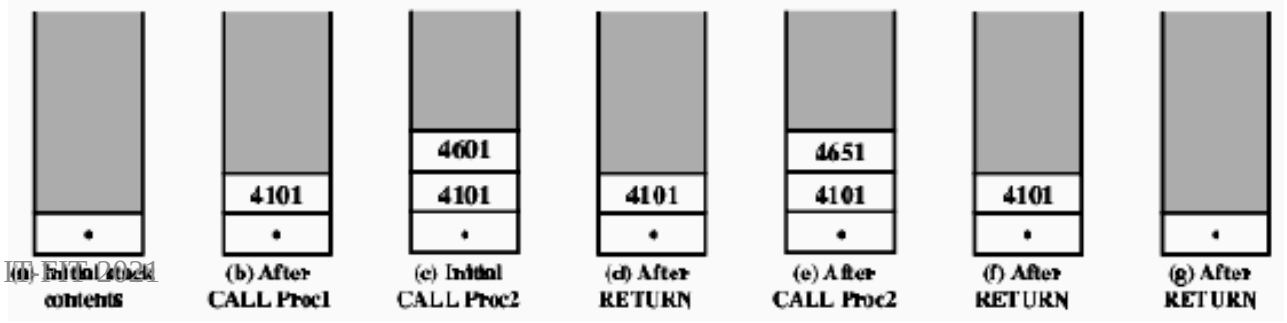
- Lệnh CALL và RETURN
  - CALL: Gọi chương trình con
    - Cất nội dung PC (chứa địa chỉ của **lệnh_kế_tiếp**) ra Stack
    - Nạp vào PC địa chỉ lệnh đầu tiên của chương trình con được gọi
    - Bộ xử lý được chuyển sang thực hiện chương trình con tương ứng
  - RETURN: Trở về từ chương trình con
    - Lấy địa chỉ của **lệnh_kế_tiếp** được cất ở Stack nạp trả lại cho PC
    - Bộ xử lý được điều khiển quay trở về thực hiện tiếp lệnh nằm sau lệnh CALL



Các kiểu thao tác của lệnh

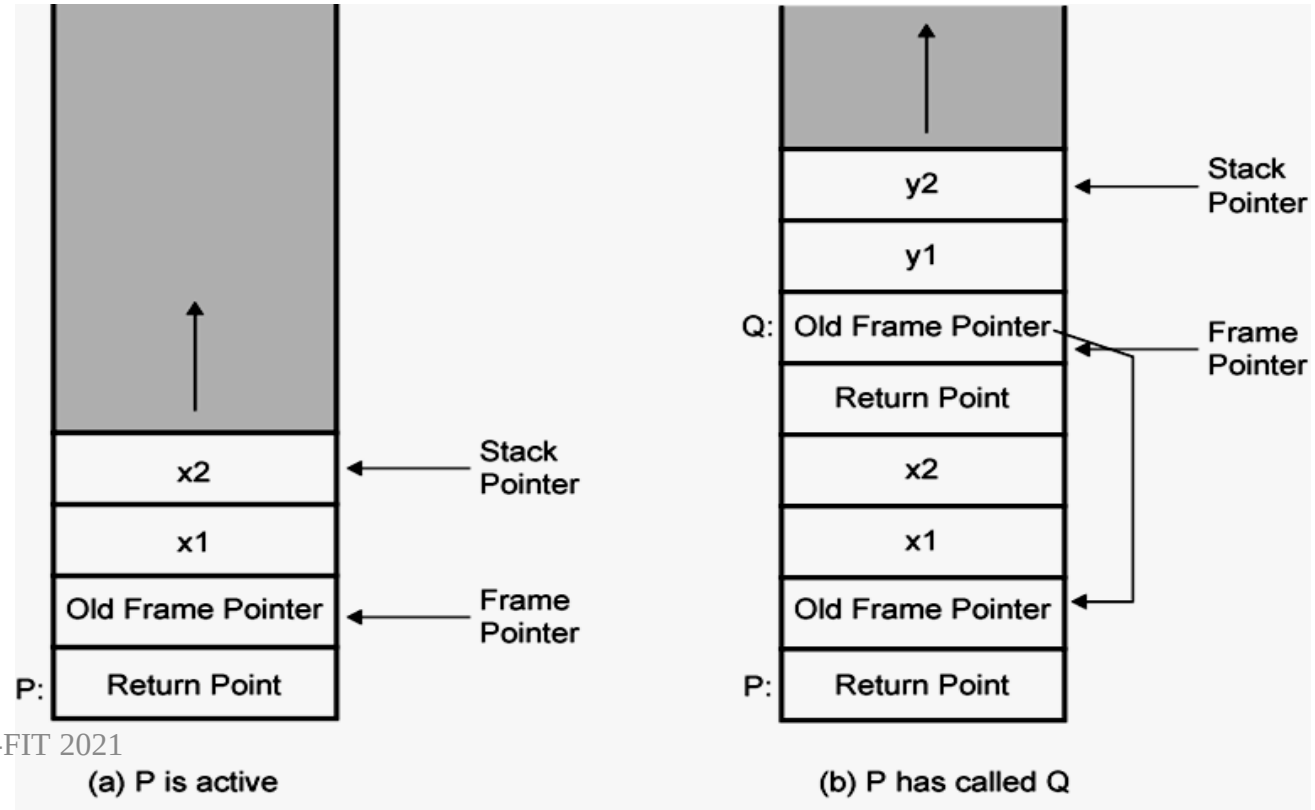
- Gọi các chương trình con lồng nhau





Các kiểu thao tác của lệnh

- Truyền tham số giữa các chương trình con
  - Truyền qua Stack
  - Ví dụ: P gọi Q(y_1,y_2) có 2 tham số.



Các kiểu thao tác của lệnh

- Các lệnh điều khiển hệ thống
  - HALT : Dừng thực hiện chương trình
  - WAIT : Tạm dừng thực hiện chương trình, lặp kiểm tra điều kiện cho đến khi thoả mãn thì tiếp tục thực hiện
  - NO OPERATION : Không thực hiện gì cả
  - LOCK : Cấm không cho xin chuyển nhượng bus
  - UNLOCK : Cho phép xin chuyển nhượng bus

Các phương pháp định địa chỉ

- Khái niệm về định địa chỉ (addressing)
  - Toán hạng của lệnh có thể là:
    - Một giá trị cụ thể nằm ngay trong lệnh
    - Nội dung của thanh ghi
    - Nội dung của ngăn nhớ hoặc cổng IO
  - Phương pháp định địa chỉ (addressing modes) là cách thức địa chỉ hóa trong vùng địa chỉ của lệnh để xác định nơi chứa toán hạng
    - Định địa chỉ tức thì
    - Định địa chỉ thanh ghi
    - Định địa chỉ trực tiếp
    - Định địa chỉ gián tiếp qua thanh ghi
    - Định địa chỉ gián tiếp
    - Định địa chỉ dịch chuyển

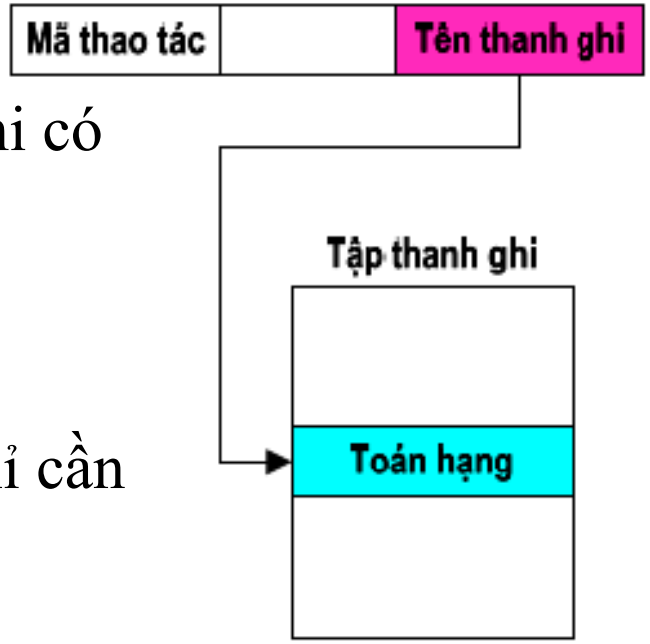
Các phương pháp định địa chỉ

- Định địa chỉ tức thì (Immediate Addressing)
  - Toán hạng nằm ngay trong vùng địa chỉ của lệnh
  - Chỉ có thể là toán hạng nguồn
  - Ví dụ: ADD R1, 5 ; R1← R1+5
  - Không tham chiếu bộ nhớ
  - Truy cập toán hạng rất nhanh
  - Dải giá trị của toán hạng bị hạn chế

Các phương pháp định địa chỉ

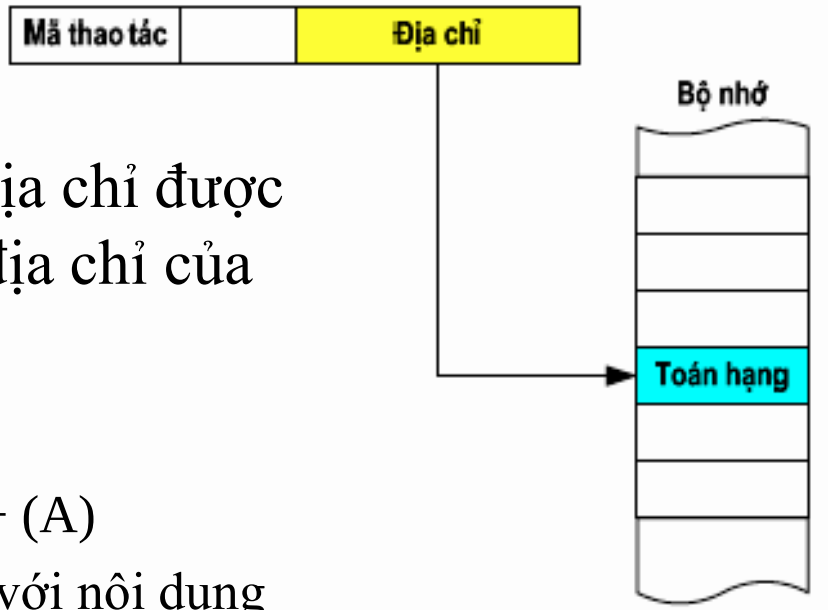
- Định địa chỉ thanh ghi (Register Addressing)
  - Toán hạng được chứa trong thanh ghi có tên trong vùng địa chỉ
  - Ví dụ:

    ADD R1, R2 ; R1← R1+R2
  - Số lượng thanh ghi ít → vùng địa chỉ cần ít bit hơn
  - Không tham chiếu bộ nhớ
  - Truy cập toán hạng nhanh
  - Tăng số lượng thanh ghi → hiệu quả hơn



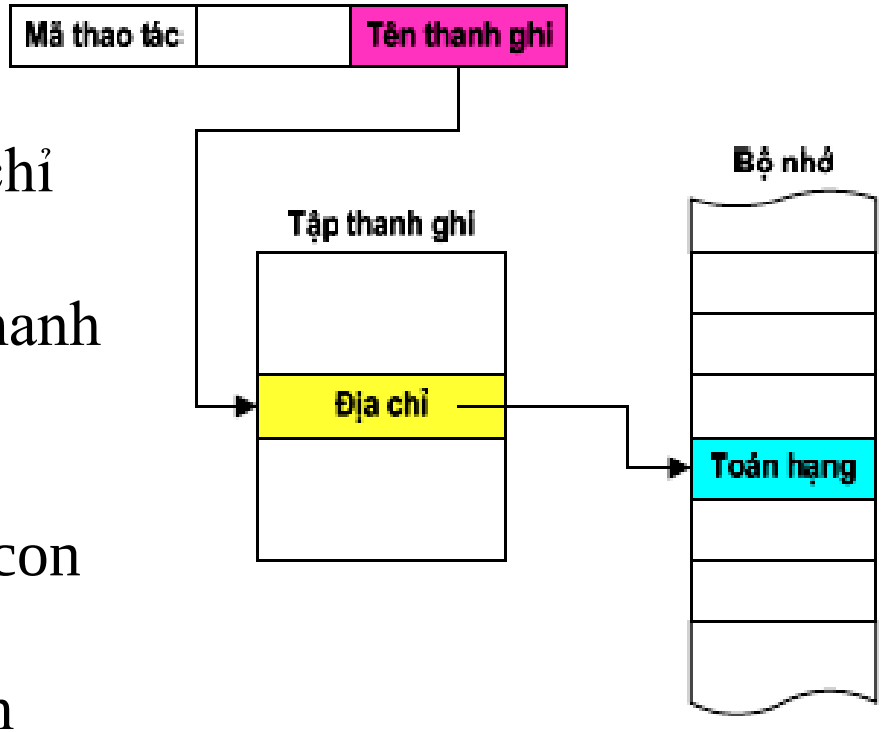
Các phương pháp định địa chỉ

- Định địa chỉ trực tiếp (Direct Addressing)
  - Toán hạng là ngăn nhớ có địa chỉ được chỉ ra trực tiếp trong vùng địa chỉ của lệnh
  - Ví dụ:
    - ADD R1, A ;R1 ← R1 + (A)
    - Cộng nội dung thanh ghi R1 với nội dung của ô nhớ có địa chỉ là A
    - Tìm toán hạng trong bộ nhớ ở địa chỉ A
  - CPU tham chiếu bộ nhớ một lần để truy nhập dữ liệu



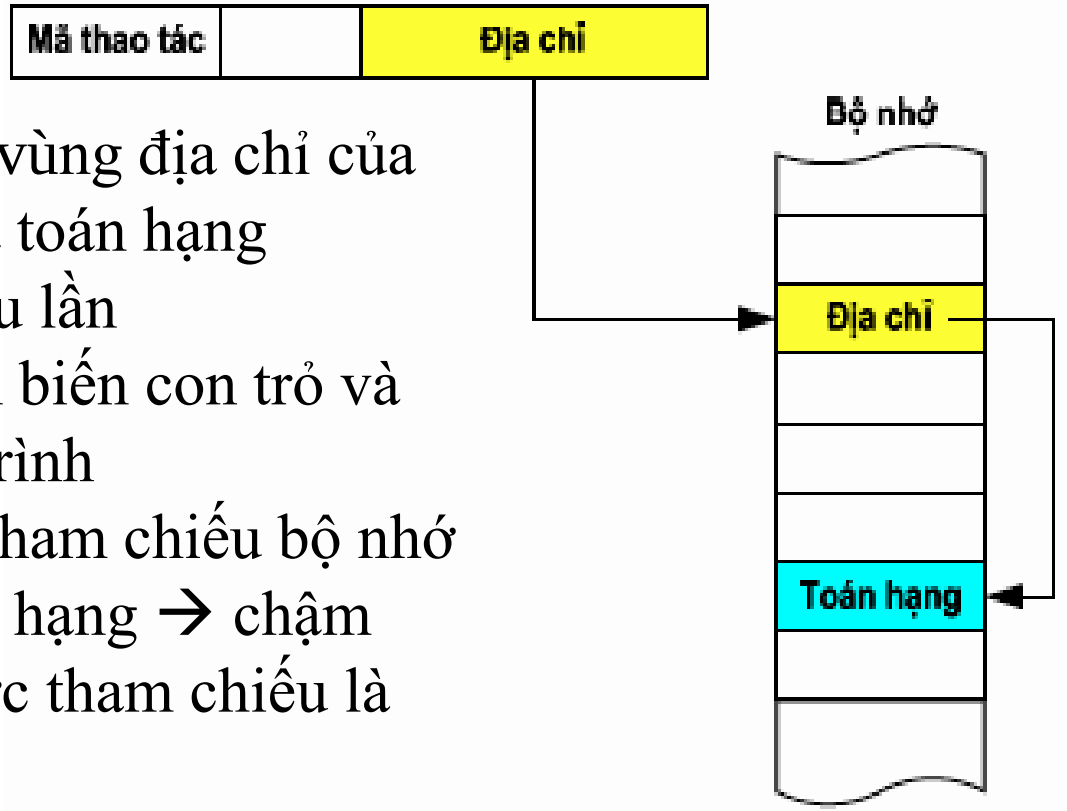
Các phương pháp định địa chỉ

- Định địa chỉ gián tiếp qua thanh ghi (Register Indirect Addressing)
  - Toán hạng là ô nhớ có địa chỉ nằm trong thanh ghi
  - Vùng địa chỉ cho biết tên thanh ghi đó. Thanh ghi có thể là ngầm định
  - Thanh ghi này được gọi là con trỏ (pointer)
  - Vùng nhớ có thể được tham chiếu là lớn (2^n, với n là độ dài của thanh ghi)



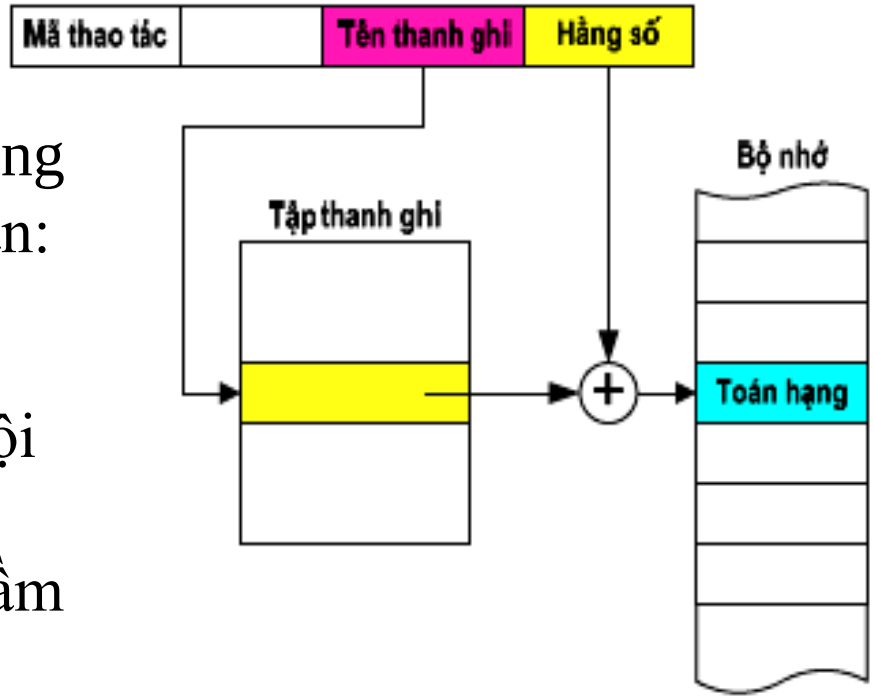
Các phương pháp định địa chỉ

- Định địa chỉ gián tiếp qua bộ nhớ (Indirect Memory Addressing)
  - Bộ nhớ được trỏ bởi vùng địa chỉ của lệnh chứa địa chỉ của toán hạng
  - Có thể gián tiếp nhiều lần
  - Giống như khái niệm biến con trỏ và biến động trong lập trình
  - CPU phải thực hiện tham chiếu bộ nhớ nhiều lần để tìm toán hạng → chậm
  - Vùng nhớ có thể được tham chiếu là lớn



Các phương pháp định địa chỉ

- Định địa chỉ dịch chuyển (Displacement Addressing)
  - Để xác định toán hạng, vùng địa chỉ chứa hai thành phần:
    - Tên thanh ghi
    - Hằng số
  - Địa chỉ của toán hạng = nội dung thanh ghi + hằng số
  - Thanh ghi có thể được ngầm định



Các phương pháp định địa chỉ

- Định địa chỉ dịch chuyển (tiếp)
  - Các dạng địa chỉ dịch chuyển
    - Địa chỉ hoá tương đối với PC
      - Thanh ghi là Bộ đếm chương trình PC
      - Toán hạng có địa chỉ cách ô nhớ được trỏ bởi PC một độ lệch xác định
    - Định địa chỉ cơ sở (base)
      - Thanh ghi chứa địa chỉ cơ sở
      - Hằng số là chỉ số
    - Định địa chỉ chỉ số (index)
      - Hằng số là địa chỉ cơ sở
      - Thanh ghi chứa chỉ số

Phân loại tập lệnh

- CISC và RISC
  - CISC:Complex Instruction Set Computer:
    - Máy tính với tập lệnh đầy đủ
    - Ví dụ: Intel x86, Motorola 680x0
  - RISC:Reduced Instruction Set Computer:
    - Máy tính với tập lệnh thu gọn
    - Ví dụ: SunSPARC, Power PC, MIPS, ARM ...
  - RISC đối nghịch với CISC

Phân loại tập lệnh

- Các đặc trưng của CISC
  - Số lượng lệnh nhiều (vài trăm lệnh) → Dễ lập trình, chương trình ngắn hơn (chiếm ít bộ nhớ)
  - Truy cập toán hạng ở các thanh ghi lẫn bộ nhớ
  - Cấu trúc CPU phức tạp
  - Thời gian thực hiện lệnh cần nhiều chu kỳ máy
  - Số lượng khuôn dạng lệnh lớn
  - CPU có tập thanh ghi nhỏ
  - Có nhiều mode địa chỉ
  - Một số lệnh không có mạch phần cứng riêng (cần có vi chương trình để thực hiện)

Phân loại tập lệnh

- Các đặc trưng của RISC
  - Số lượng lệnh ít (vài chục lệnh) và cơ bản nhất → Khó lập trình, chương trình dài hơn
  - Hầu hết các lệnh truy cập toán hạng ở các thanh ghi
  - Cấu trúc CPU đơn giản
  - Thời gian thực hiện lệnh là một chu kỳ máy
  - Số lượng khuôn dạng lệnh ít (<=4)
  - CPU có tập thanh ghi lớn
  - Có ít mode địa chỉ (<=4)
  - Mỗi lệnh có mạch phần cứng riêng (không cần vi chương trình)

Phân loại tập lệnh

- So sánh CISC và RISC

Loại	CISC			RISC	
Hãng SX	IBM	DEC VAX	Intel	Motorola	MIPS
Hệ thống MT	370/168	11/780	486	88000	R4000
Năm SX	1973	1978	1989	1988	1991
Số lượng lệnh	208	303	235	51	94
Kích thước lệnh (B)	2-6	2-57	1-11	4	32
Addressing modes	4	22	11	3	1
Số lượng thanh ghi	16	16	8	32	32
Vi ChươngTrình (KB)	420	480	246	0	0

Phân loại tập lệnh

- Thống kê 10 lệnh Intel x86 sử dụng nhiều nhất

TT	Lệnh	Tỷ lệ (%)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

Phân loại tập lệnh

- Tại sao kiến trúc CISC của Intel vẫn sử dụng nhiều?
  - Vấn đề tương thích
  - Dễ xây dựng trình dịch (compiler) hơn
  - Phù hợp với nhiều NNLT cấp cao (HLL)
  - Phần mềm có sẵn đang sử dụng nhiều
  - Thực tế hiện nay sử dụng hệ thống tập lệnh lai giữa RISC và CISC
    - Tổ chức bên trong theo RISC
    - Kiến trúc lập trình bên ngoài theo CISC
    - Sử dụng vi chương trình làm trung gian

Phân loại tập lệnh

- Ưu nhược điểm của CISC
  - Ưu điểm
    - Chương trình ít lệnh hơn, ít tốn bộ nhớ để lưu trữ
    - Truy cập bộ nhớ với ít lệnh hơn
    - Chương trình dễ viết, dễ đọc và dễ hiểu hơn
  - Nhược điểm
    - Dạng lệnh phức tạp, giải mã lệnh chậm
    - Lệnh phức tạp nên không uyển chuyển, không áp dụng cho nghiều trường hợp khác nhau
    - Xử lý ngắt chậm hơn (do lệnh chiếm nhiều chu kỳ máy) nên thời gian đáp ứng kém

Kiến trúc tập lệnh Intel x86

Intel Processor	Date of Product Intro-duction	Perform-ance in MIPs[1]	Max. CPU Frequency at Intro-duction	No. of Transis-tors on the Die	Main CPU Register Size[2]	Extern. Data Bus Size[2]	Max. Extern. Addr. Space	Caches in CPU Pack-age[3]
8086	1978	0.8	8 MHz	29 K	16	16	1 MB	None
Intel 286	1982	2.7	12.5 MHz	134 K	16	16	16 MB	Note 3
Intel386™ DX	1985	6.0	20 MHz	275 K	32	32	4 GB	Note 3
Intel486™ DX	1989	20	25 MHz	1.2 M	32	32	4 GB	8KB L1
Pentium®	1993	100	60 MHz	3.1 M	32	64	4 GB	16KB L1
Pentium® Pro	1995	440	200 MHz	5.5 M	32	64	64 GB	16KB L1; 256KB or 512KB L2
Pentium II®	1997	466	266	7 M	32	64	64 GB	32KB L1; 256KB or 512KB L2
Pentium® III	1999	1000	500	8.2 M	32 GP 128 SIMD-FP	64	64 GB	32KB L1; 512KB L2

Kiến trúc tập lệnh Intel x86

Intel Processor	Date Intro-duced	Microarhi-tecture	Clock Frequency at Intro-duction	Tran-sistors Per Die	Register Sizes[1]	System Bus Band-width	Max. Extern. Addr. Space	On-Die Caches[2]
Pentium 4 Processor	2000	Intel NetBurst Microarchi-tecture	1.50 GHz	42 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Execution Trace Cache; 8KB L1; 256-KB L2
Intel Xeon Processor	2001	Intel NetBurst Microarchi-tecture	1.70 GHz	42 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Trace Cache; 8-KB L1; 256-KB L2
Intel Xeon Processor	2002	Intel NetBurst Microarchi-tecture; Hyper-Threading Technology	2.20 GHz	55 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Trace Cache; 8-KB L1; 512-KB L2

Kiến trúc tập lệnh Intel x86

Intel Processor	Date Intro-duced	Microarhi-tecture	Clock Frequency at Intro-duction	Tran-sistors Per Die	Register Sizes[1]	System Bus Band-width	Max. Extern. Addr. Space	On-Die Caches[2]
Intel Pentium 4 Processor Supporting Hyper-Threading Technology at 90 nm process	2004	Intel NetBurst Microarchi-tecture; Hyper-Threading Technology	3.40 GHz	125 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	6.4 GB/s	64 GB	12K µop Execution Trace Cache; 16 KB L1; 1 MB L2
Intel Pentium M Processor 755[3]	2004	Intel Pentium M Processor	2.00 GHz	140 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	4 GB	L1: 64 KB L2: 2MB

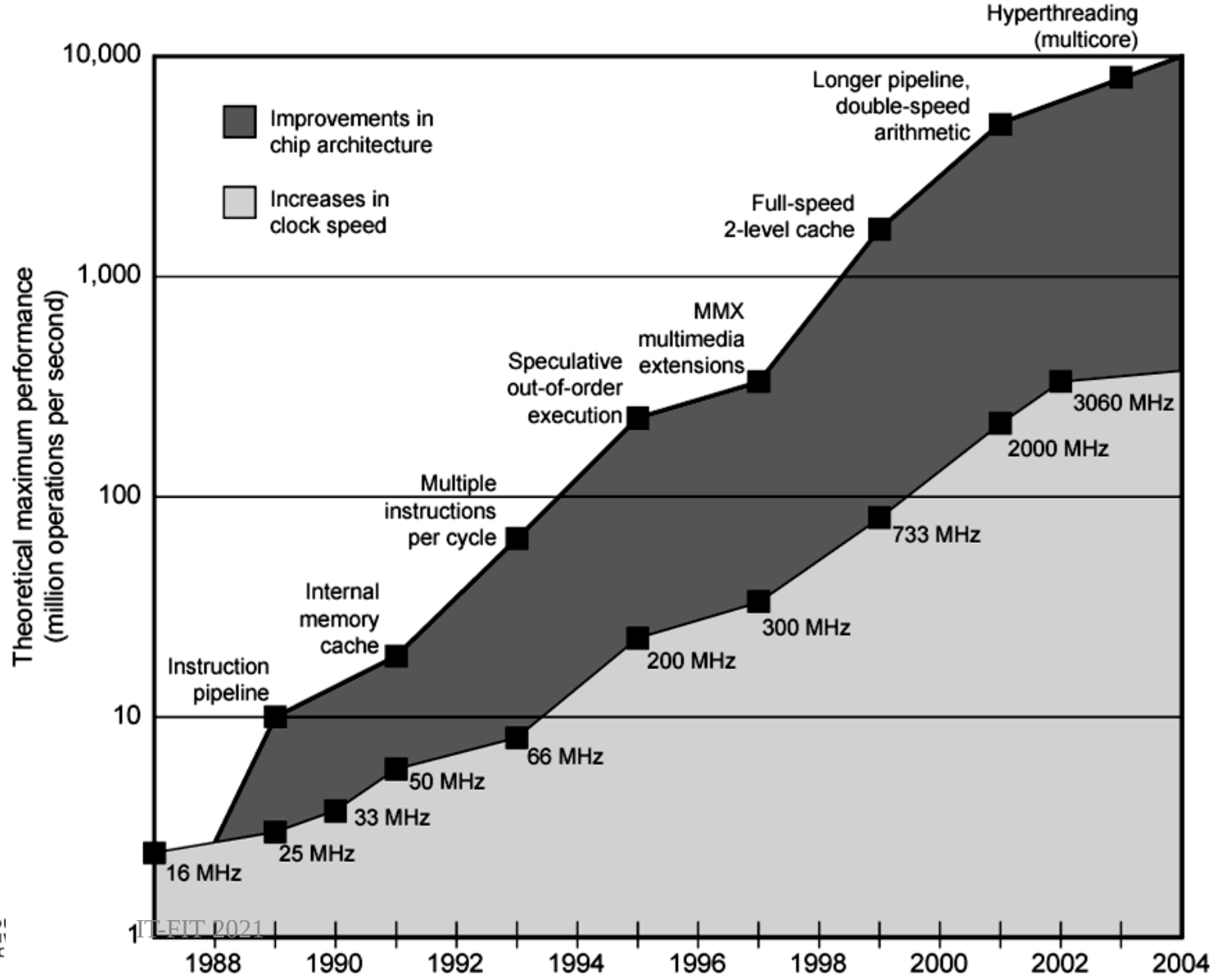
Kiến trúc tập lệnh Intel x86

Intel Processor	Date Intro-duced	Micro-architec-ture	Top-Bin Fre-quency at Intro-duction	Tran-sistor s	Register Sizes	System Bus/QP I Link Speed	Max. Extern . Addr. Space	On-Die Caches
Quad-core Intel Xeon Processor 5355	2006	Intel Core Microarchitecture; Quad Core; Intel 64 Architecture; Intel Virtualization Technology.	2.66 GHz	582 M	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128	10.6 GB/s	256 GB	L1: 64 KB L2: 4MB (8 MB Total)
Intel Core 2 Duo Processor E6850	2007	Intel Core Microarchitecture; Dual Core; Intel 64 Architecture; Intel Virtualization Technology; Intel Trusted Execution Technology	3.00 GHz	291 M	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128	10.6 GB/s	64 GB	L1: 64 KB L2: 4MB (4MB Total)
Intel Xeon Processor 7350	2007	Intel Core Microarchitecture; Quad Core; Intel 64 Architecture; Intel Virtualization Technology.	2.93 GHz	582 M	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128	8.5 GB/s	1024 GB	L1: 64 KB L2: 4MB (8MB Total)
Intel Xeon Processor 5472	2007	Enhanced Intel Core Microarchitecture; Quad Core; Intel 64 Architecture; Intel Virtualization Technology.	3.00 GHz	820 M	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128	12.8 GB/s	256 GB	L1: 64 KB L2: 6MB (12MB Total)

Kiến trúc tập lệnh Intel x86

Intel Processor	Date Intro-duced	Micro-architec-ture	Top-Bin Fre-quency at Intro-duction	Tran-sistor s	Register Sizes	System Bus/QP I Link Speed	Max. Extern . Addr. Space	On-Die Caches
Intel Core i7-620M Processor	2010	Intel Turbo Boost Technology, Intel microarchitecture codename Westmere; Dualcore; HyperThreading Technology; Intel 64 Architecture; Intel Virtualization Technology., Integrated graphics	2.66 GHz	383 M	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128		64 GB	L1: 64 KB L2: 256KB L3: 4MB
Intel Xeon-Processor 7560	2010	Intel Turbo Boost Technology, Intel microarchitecture codename Nehalem; Eight core; HyperThreading Technology; Intel 64 Architecture; Intel Virtualization Technology.	2.26 GHz	2.3B	GP: 32, 64 FPU: 80 MMX: 64 XMM: 128	QPI: 6.4 GT/s; Memory: 50 GB/s	16 TB	L1: 64 KB L2: 256KB L3: 24MB

Kiến trúc tập lệnh Intel x86



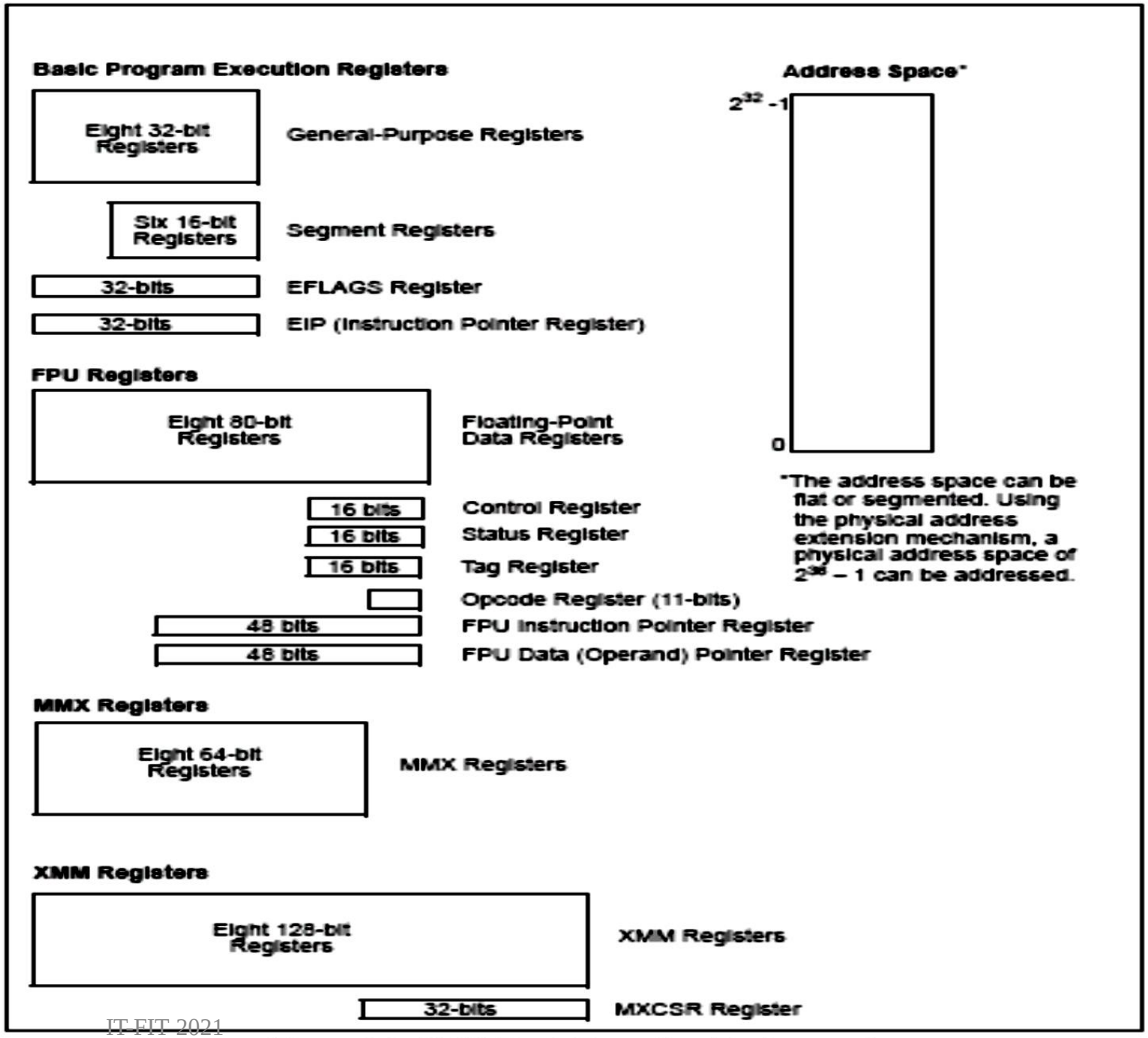


Figure 3-1. IA-32 Basic Execution Environment

Kiến trúc tập lệnh Intel x86

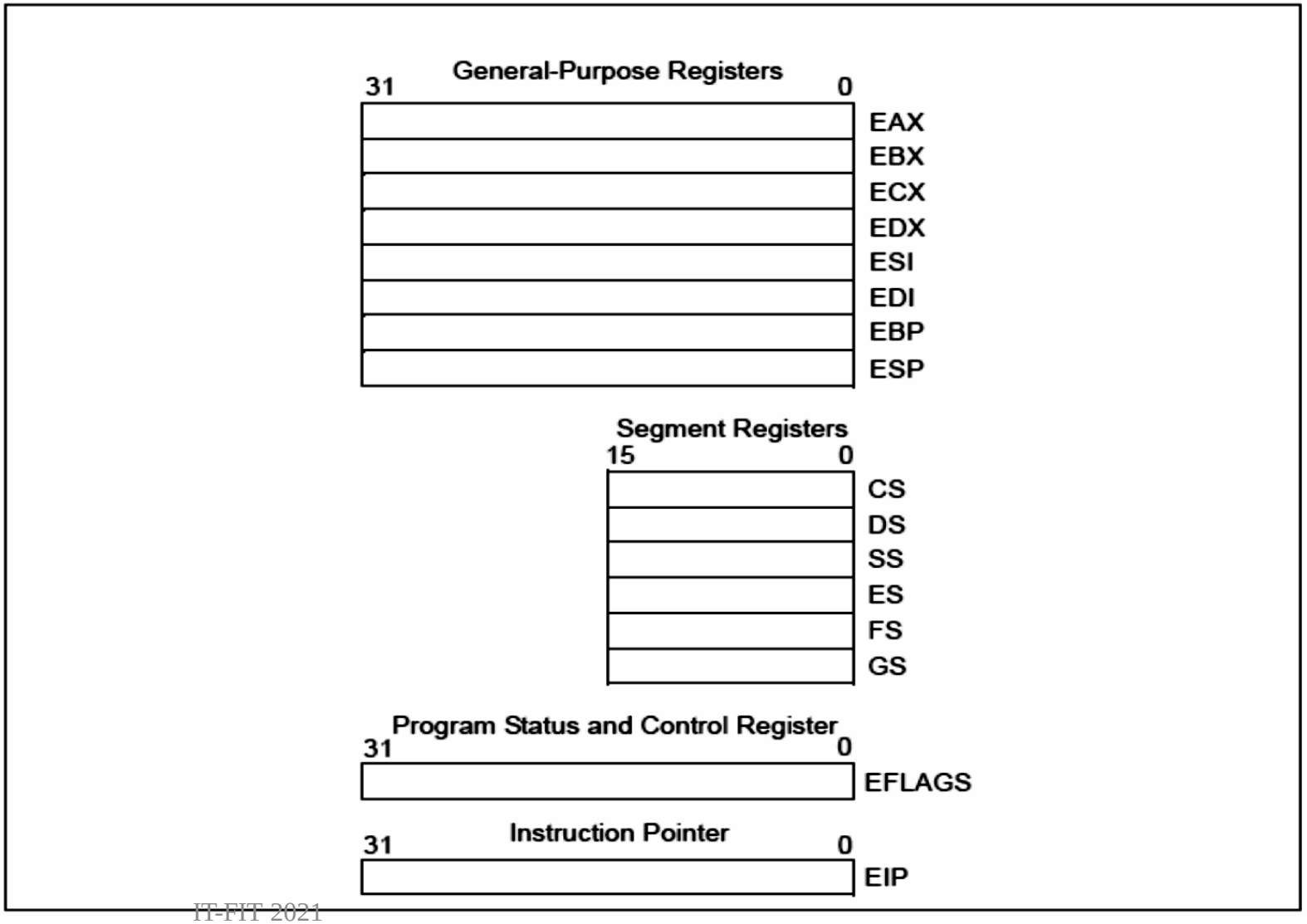


Figure 3-3. General System and Application Programming Registers

Kiến trúc tập lệnh Intel x86

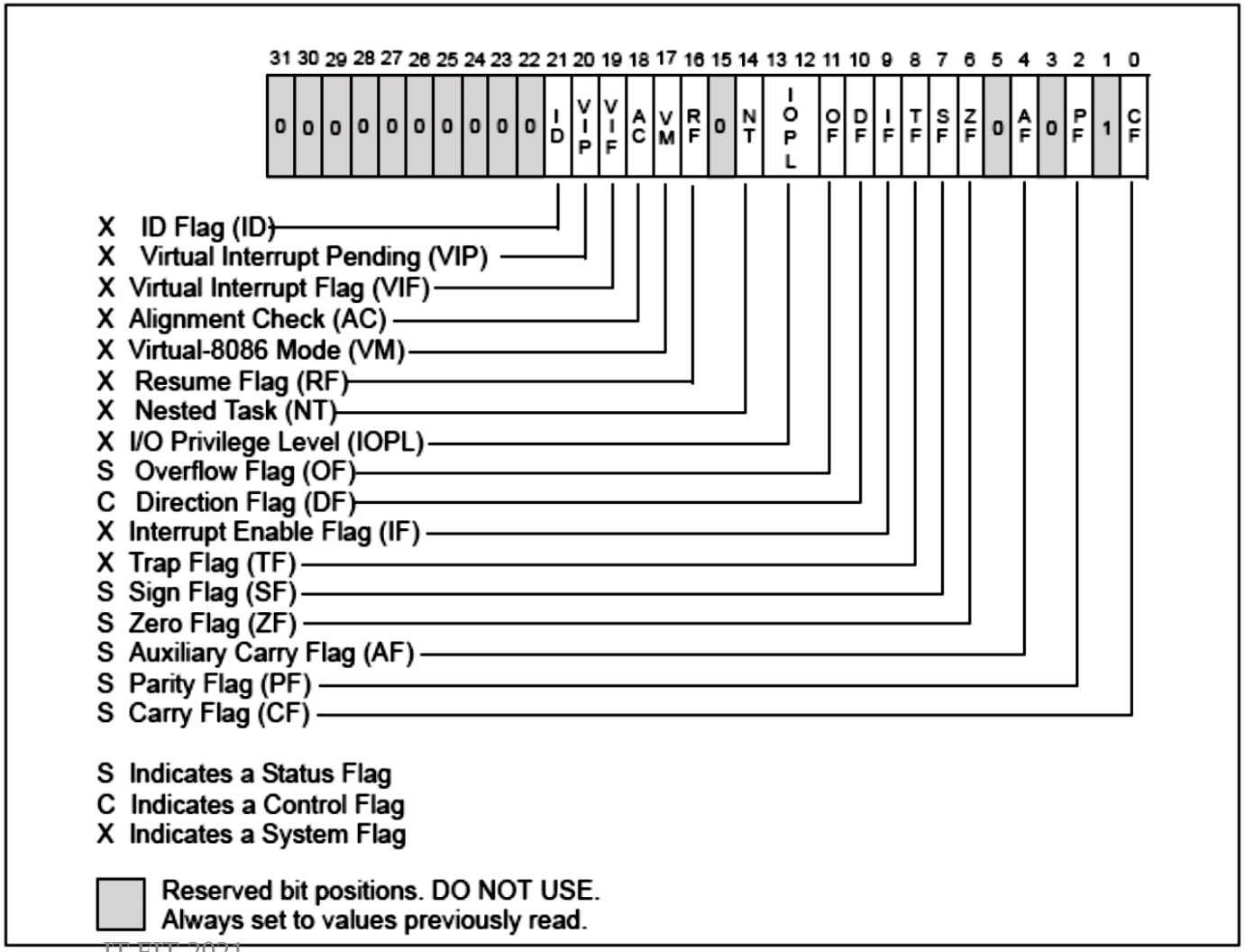


Figure 3-7. EFLAGS Register

Kiến trúc tập lệnh Intel x86

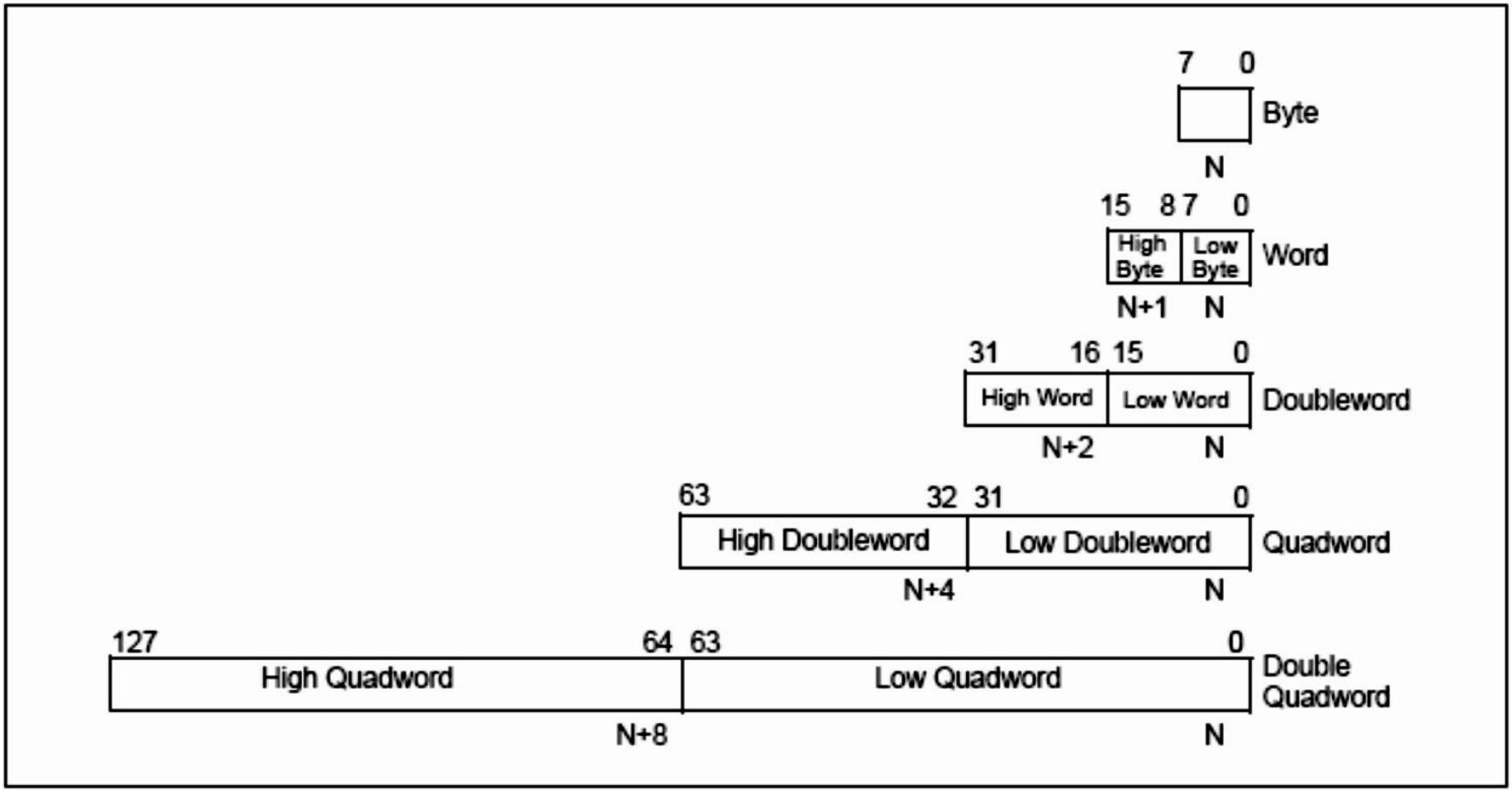


Figure 4-1. Fundamental Data Types

Kiến trúc tập lệnh Intel x86

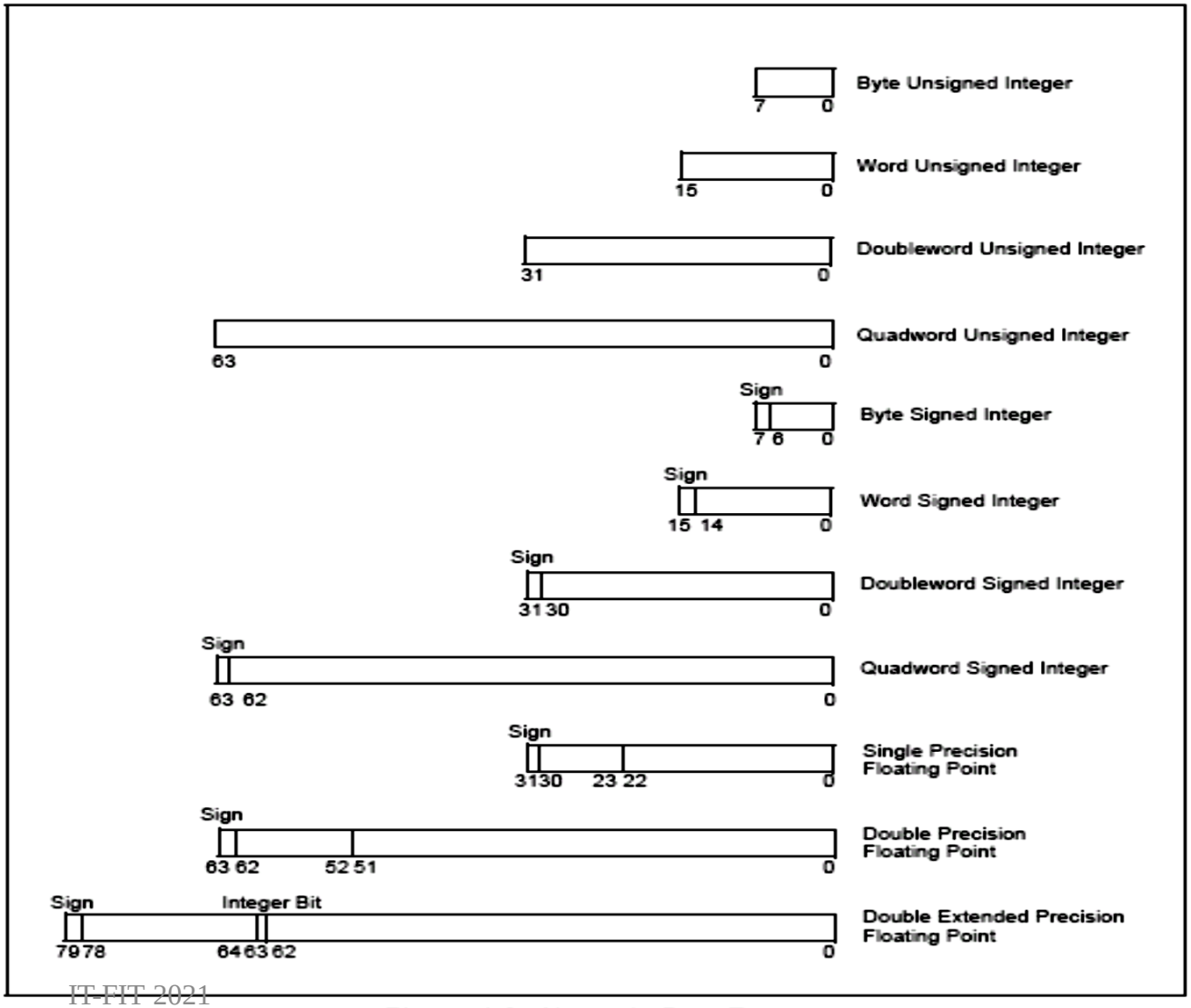


Figure 4-3. Numeric Data Types

Kiến trúc tập lệnh Intel x86

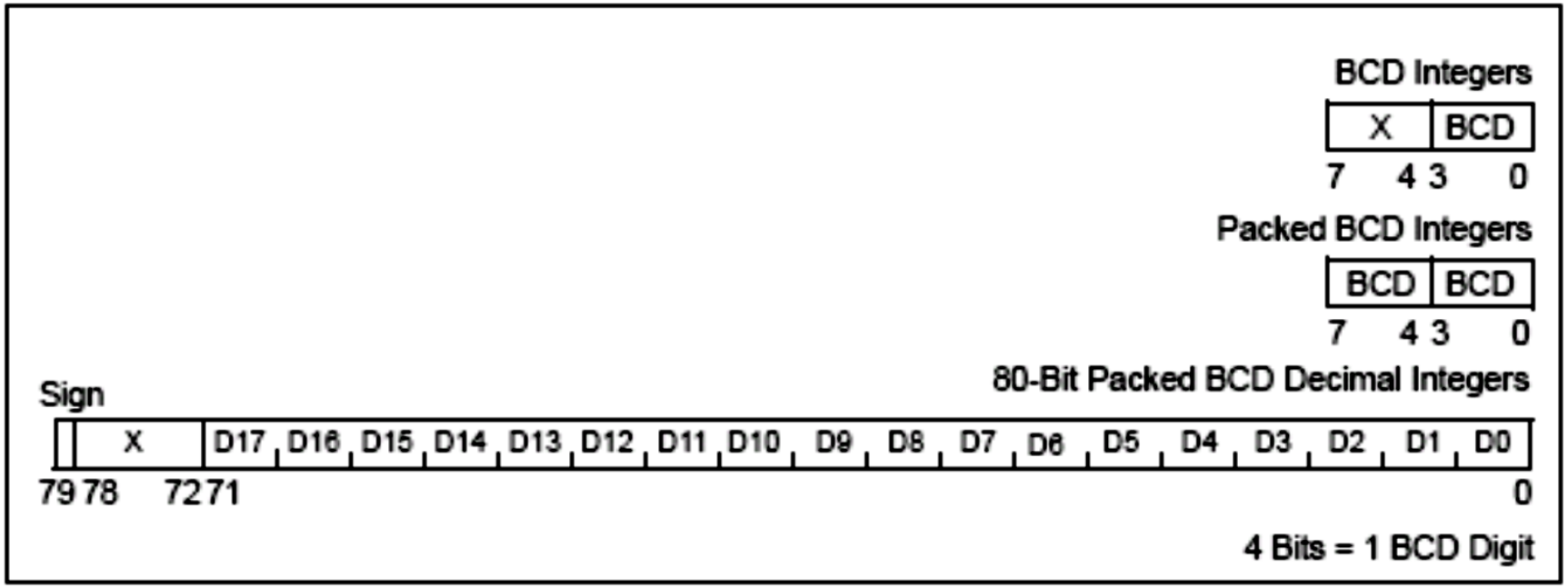


Figure 4-8. BCD Data Types

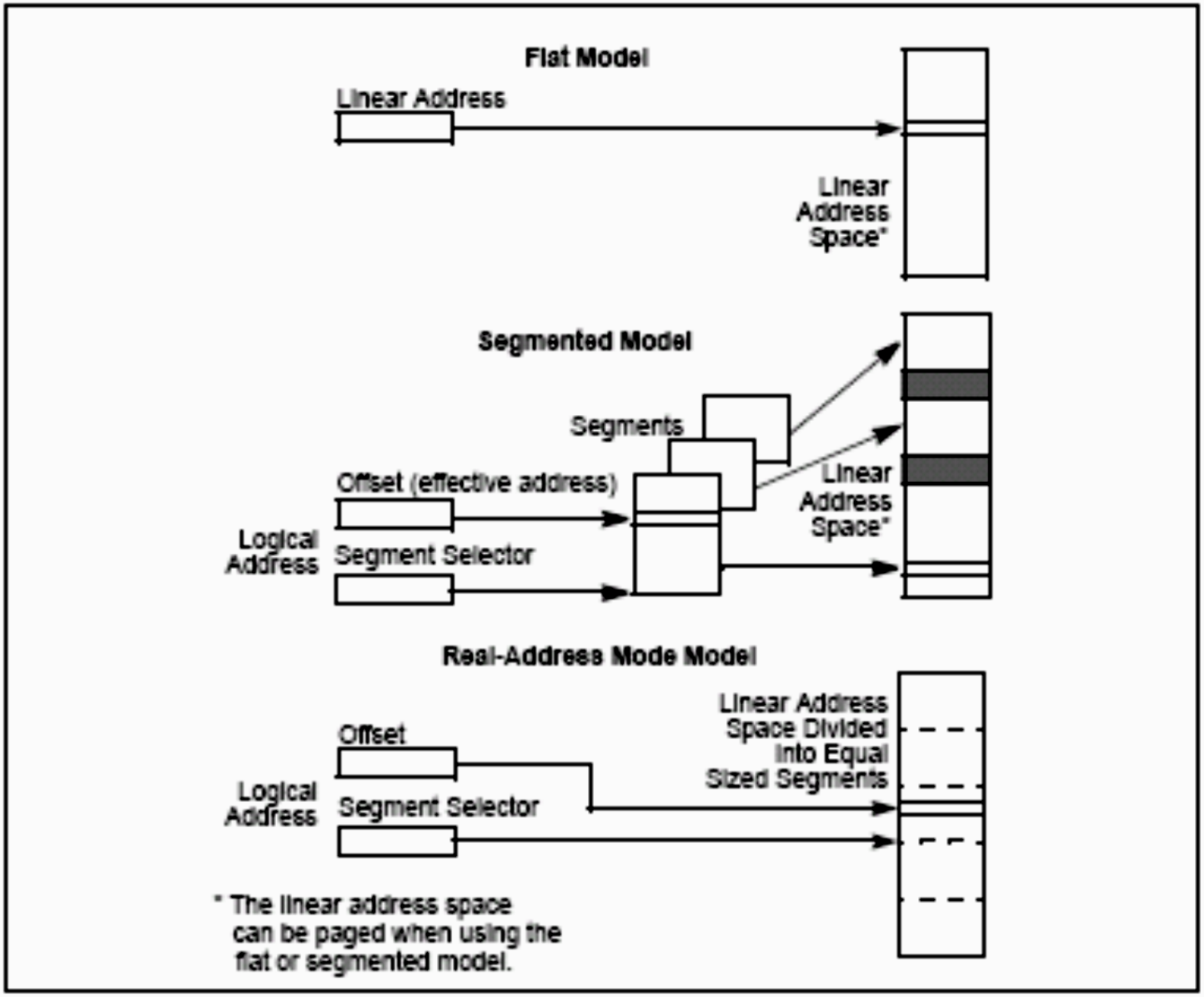


Figure 3-2. Three Memory Management Models

Kiến trúc tập lệnh Intel x86

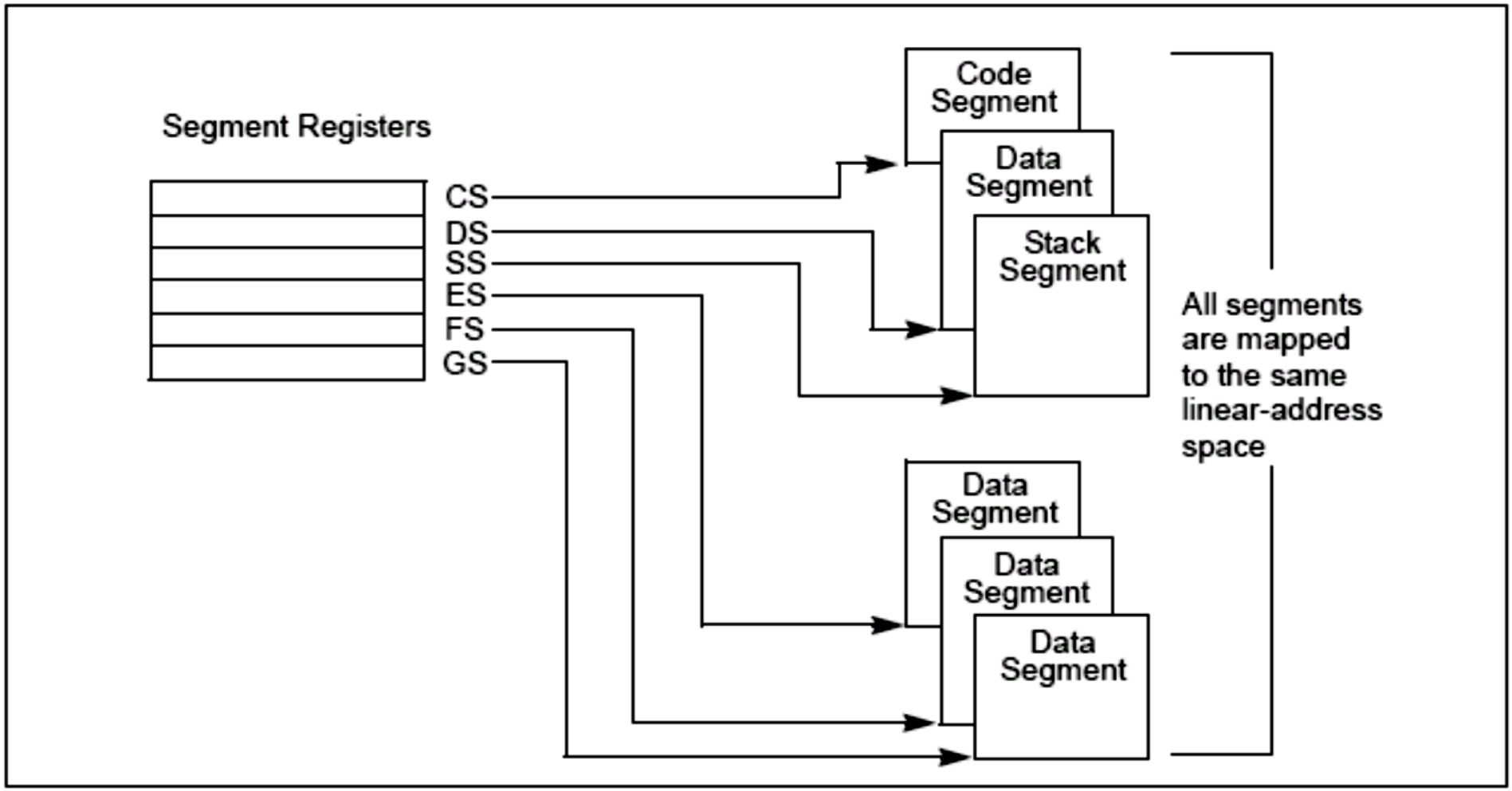


Figure 3-6. Use of Segment Registers in Segmented Memory Model

Kiến trúc tập lệnh Intel x86

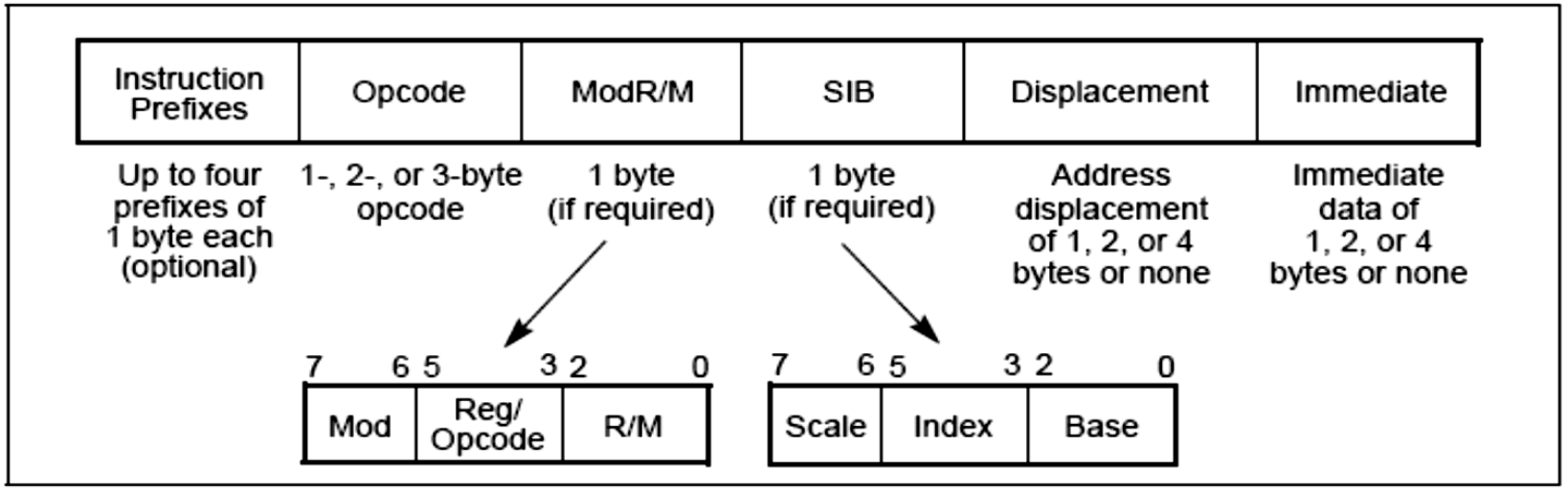


Figure 2-1. IA-32 Instruction Format

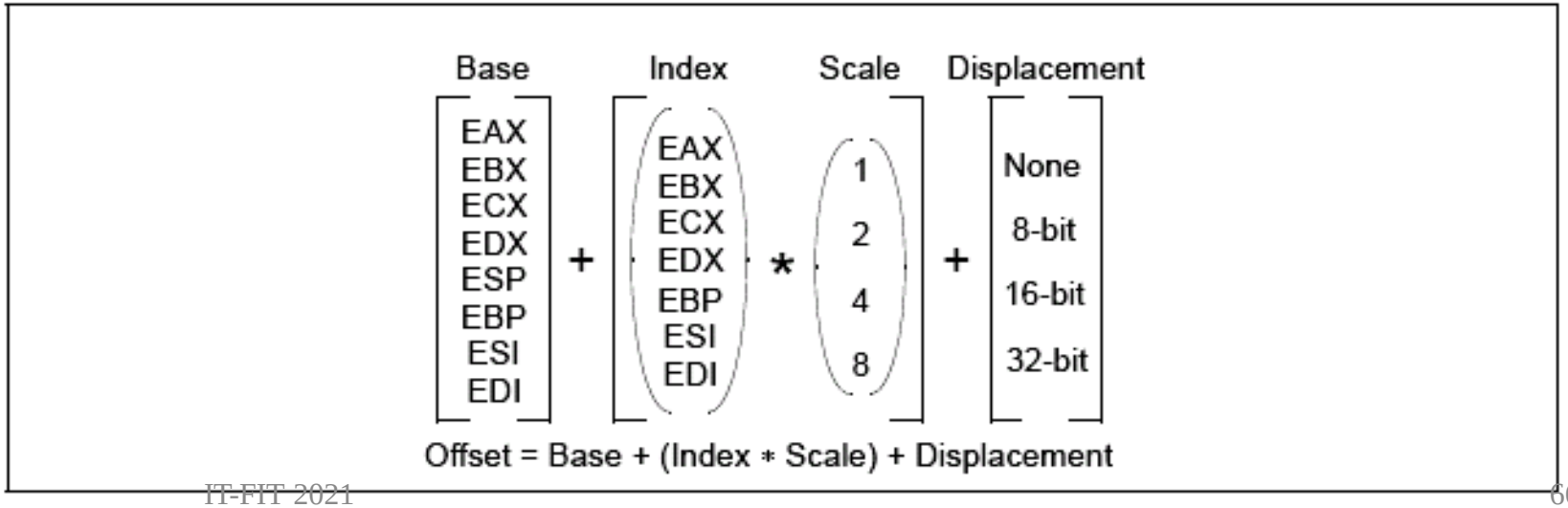
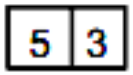
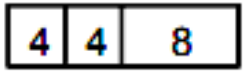
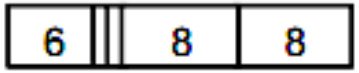
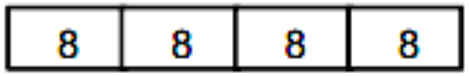
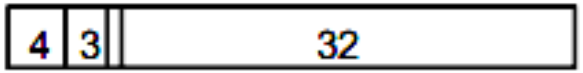


Figure 3-9. Offset (or Effective Address) Computation

Kiến trúc tập lệnh Intel x86

- **Ví dụ về khuôn dạng lệnh của CPU Intel x86**

Type	Format (field widths shown)	Opcode	Description of operand(s)
1-byte		PUSH	3-bit register specification
2-byte		JE	4-bit condition, 8-bit jump offset
3-byte		MOV	8-bit register/mode, 8-bit offset
4-byte		XOR	8-bit register/mode, 8-bit base/index, 8-bit offset
5-byte		ADD	3-bit register spec, 32-bit immediate
6-byte		TEST	8-bit register/mode, 32-bit immediate

Kiến trúc tập lệnh Intel x86

Moves

MOV DST,SRC	Move SRC to DST
PUSH SRC	Push SRC onto the stack
POP DST	Pop a word from the stack to DST
XCHG DS1,DS2	Exchange DS1 and DS2
LEA DST,SRC	Load effective addr of SRC into DST
CMOVcc DST,SRC	Conditional move

Arithmetic

ADD DST,SRC	Add SRC to DST
SUB DST,SRC	Subtract SRC from DST
MUL SRC	Multiply EAX by SRC (unsigned)
IMUL SRC	Multiply EAX by SRC (signed)
DIV SRC	Divide EDX:EAX by SRC (unsigned)
IDIV SRC	Divide EDX:EAX by SRC (signed)
ADC DST,SRC	Add SRC to DST, then add carry bit
SBB DST,SRC	Subtract SRC & carry from DST
INC DST	Add 1 to DST
DEC DST	Subtract 1 from DST
NEG DST	Negate DST (subtract it from 0)

Binary coded decimal

DAA	Decimal adjust
DAS	Decimal adjust for subtraction
AAA	ASCII adjust for addition
AAS	ASCII adjust for subtraction
AAM	ASCII adjust for multiplication
AAD	ASCII adjust for division

Boolean

AND DST,SRC	Boolean AND SRC into DST
OR DST,SRC	Boolean OR SRC into DST
XOR DST,SRC	Boolean Exclusive OR SRC to DST
NOT DST	Replace DST with 1's complement

Shift/rotate

SAL/SAR DST,#	Shift DST left/right # bits
SHL/SHR DST,#	Logical shift DST left/right # bits
ROL/ROR DST,#	Rotate DST left/right # bits
RCL/RCR DST,#	Rotate DST through carry # bits

Kiến trúc tập lệnh Intel x86

Transfer of control

JMP ADDR	Jump to ADDR
Jxx ADDR	Conditional jumps based on flags
CALL ADDR	Call procedure at ADDR
RET	Return from procedure
IRET	Return from interrupt
LOOPxx	Loop until condition met
INT n	Initiate a software interrupt
INTO	Interrupt if overflow bit is set

Strings

LODS	Load string
STOS	Store string
MOVS	Move string
CMPS	Compare two strings
SCAS	Scan Strings

Test/compare

TEST SRC1,SRC2	Boolean AND operands, set flags
CMP SRC1,SRC2	Set flags based on SRC1 - SRC2

Miscellaneous

SWAP DST	Change endianness of DST
CWQ	Extend EAX to EDX:EAX for division
CWDE	Extend 16-bit number in AX to EAX
ENTER SIZE,LV	Create stack frame with SIZE bytes
LEAVE	Undo stack frame built by ENTER
NOP	No operation
HLT	Halt
IN AL,PORT	Input a byte from PORT to AL
OUT PORT,AL	Output a byte from AL to PORT
WAIT	Wait for an interrupt

SRC = source
DST = destination
# = shift/rotate count
LV = # locals

Kiến trúc tập lệnh Intel x86

ADC—Add with Carry

Opcode	Instruction	Description
14 *ib*	ADC AL,*imm8*	Add with carry *imm8* to AL
15 *iw*	ADC AX,*imm16*	Add with carry *imm16* to AX
15 *id*	ADC EAX,*imm32*	Add with carry *imm32* to EAX
80 /2 *ib*	ADC *r/m8,imm8*	Add with carry *imm8* to *r/m8*
81 /2 *iw*	ADC *r/m16,imm16*	Add with carry *imm16* to *r/m16*
81 /2 *id*	ADC *r/m32,imm32*	Add with CF *imm32* to *r/m32*
83 /2 *ib*	ADC *r/m16,imm8*	Add with CF sign-extended *imm8* to *r/m16*
83 /2 *ib*	ADC *r/m32,imm8*	Add with CF sign-extended *imm8* into *r/m32*
10 /r	ADC *r/m8,r8*	Add with carry byte register to *r/m8*
11 /r	ADC *r/m16,r16*	Add with carry *r16* to *r/m16*
11 /r	ADC *r/m32,r32*	Add with CF *r32* to *r/m32*
12 /r	ADC *r8,r/m8*	Add with carry *r/m8* to byte register
13 /r	ADC *r16,r/m16*	Add with carry *r/m16* to *r16*
13 /r	ADC *r32,r/m32*	Add with CF *r/m32* to *r32*

Câu hỏi