TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG & TIN HỌC

Bài giảng

KIẾN TRÚC MÁY TÍNH

Giảng viên : Phạm Huyền Linh
Bộ môn : Toán Tin

CHƯƠNG 5

BỘ XỬ LÝ

(Central Processing Unit)

Giảng viên: Phạm Huyền Linh

Chương 6

1. Tổ chức của CPU
2. Thiết kế đơn vị điều khiển
3. Kỹ thuật đường ống lệnh

1. Tổ chức của CPU

- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi (Register)
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Nhiệm vụ của CPU

- Nhận lệnh (Fetch Instruction):

  CPU đọc lệnh từ bộ nhớ

- Giải mã lệnh (Decode Instruction):

  Xác định thao tác mà lệnh yêu cầu

- Nhận dữ liệu (Fetch Data):

  Quá trình thực hiện lệnh có thể đòi hỏi dữ liệu từ bộ nhớ hoặc I/O sytsem

- Xử lý dữ liệu (Process Data):

  Quá trình thực hiện lệnh có thể yêu cầu tính toán số học hay logic

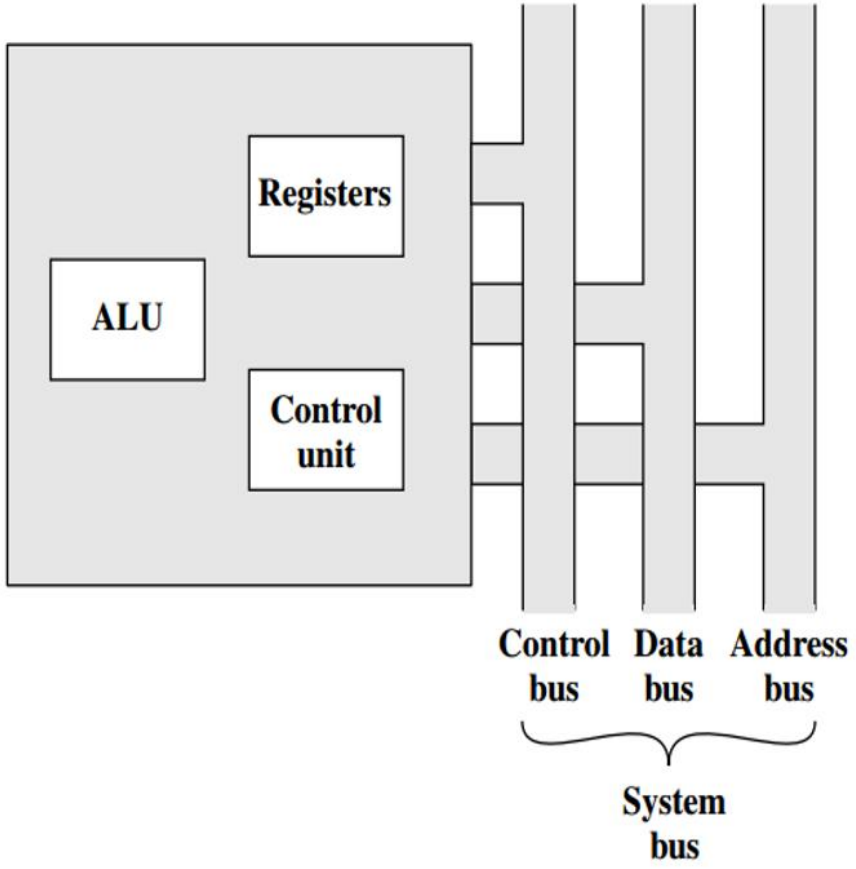
- Ghi dữ liệu (Write Data):

  Kết quả của lệnh có thể ghi ra ngoài bộ nhớ, I/O system

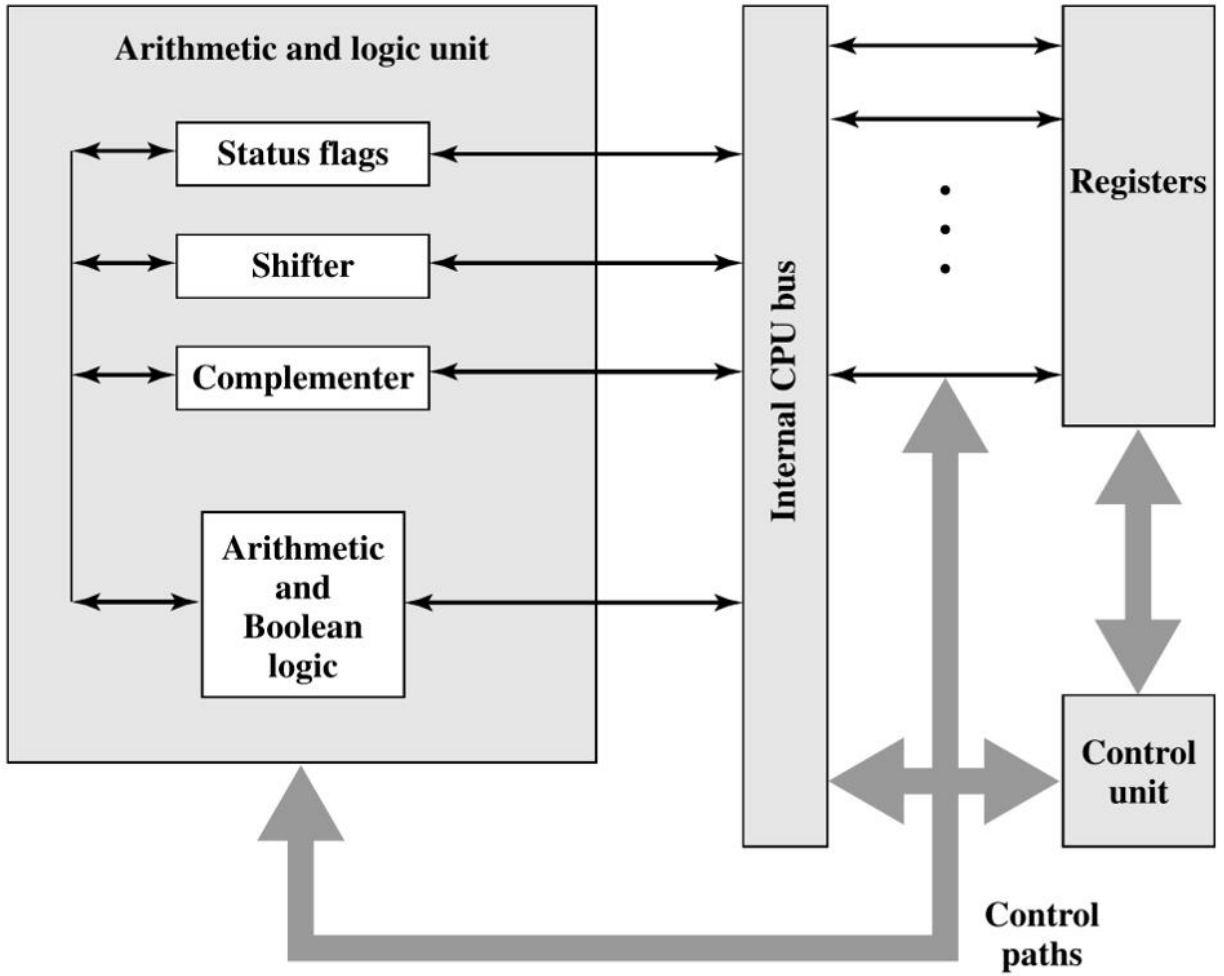
1. Tổ chức của CPU

- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi (Register)
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Sơ đồ cấu tạo



The CPU with the System Bus



Internal Structure of the CPU

6.1. Tổ chức của CPU

- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi (Register)
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Thanh ghi

- User-visible registers
  - General purpose (Thanh ghi với mục đích chung)
  - Data (Thanh ghi dữ liệu)
  - Address (Thanh ghi địa chỉ)
  - Condition codes (Thanh ghi mã điều kiện): Sign, Zero, Carry, equal, Overflow, Interrupt
- Control and status registers
  - Program counter (PC): Chứa địa chỉ của lệnh được nạp tiếp theo
  - Instruction register (IR): Chứa lệnh đang thực hiện
  - Memory address register (MAR): Chứa địa chỉ của một ô nhớ
  - Memory buffer register (MBR): Chứa từ nhớ của Data được ghi vào bộ nhớ hay được nạp gần nhất.

1. Tổ chức của CPU

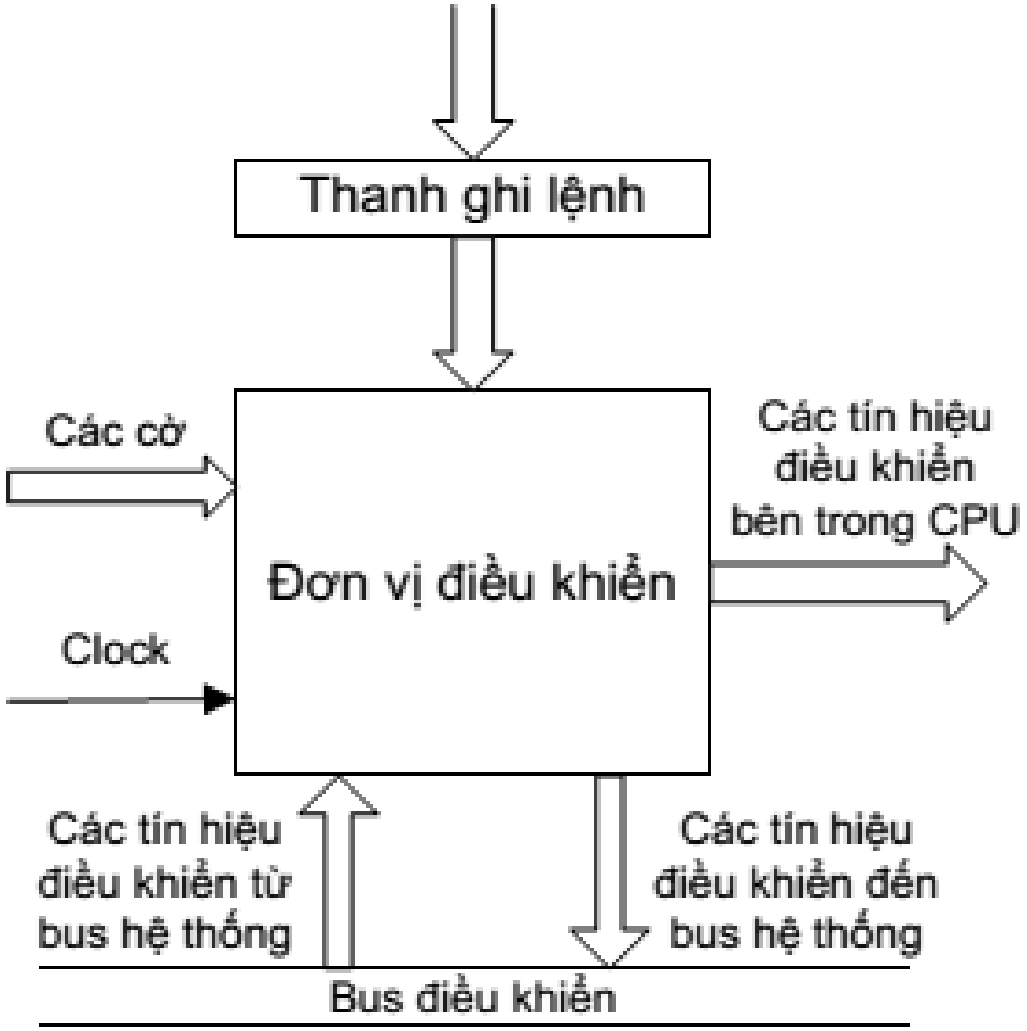
- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Đơn vị điều khiển

- Chức năng
  - Điều khiển nhận lệnh từ bộ nhớ đưa vào CPU
  - Tăng nội dung của PC để trỏ sang lệnh kế tiếp
  - Giải mã lệnh đã được nhận để xác định thao tác mà lệnh yêu cầu
  - Phát ra các tín hiệu điều khiển thực hiện lệnh
  - Nhận các tín hiệu yêu cầu từ bus điều khiển và đáp ứng với các yêu cầu đó.

Mô hình kết nối đơn vị điều khiển

- Tín hiệu vào:
  - Clock: tín hiệu nhịp từ mạch tạo dao động bên ngoài
  - Lệnh từ thanh ghi lệnh đưa đến để giải mã
  - Các cờ từ thanh ghi cờ cho biết trạng thái của CPU
  - Các tín hiệu yêu cầu từ bus điều khiển
- Tín hiệu ra:
  - Điều khiển bên trong CPU: ALU, Registers
  - Điều khiển bên ngoài: Memory, I/O system

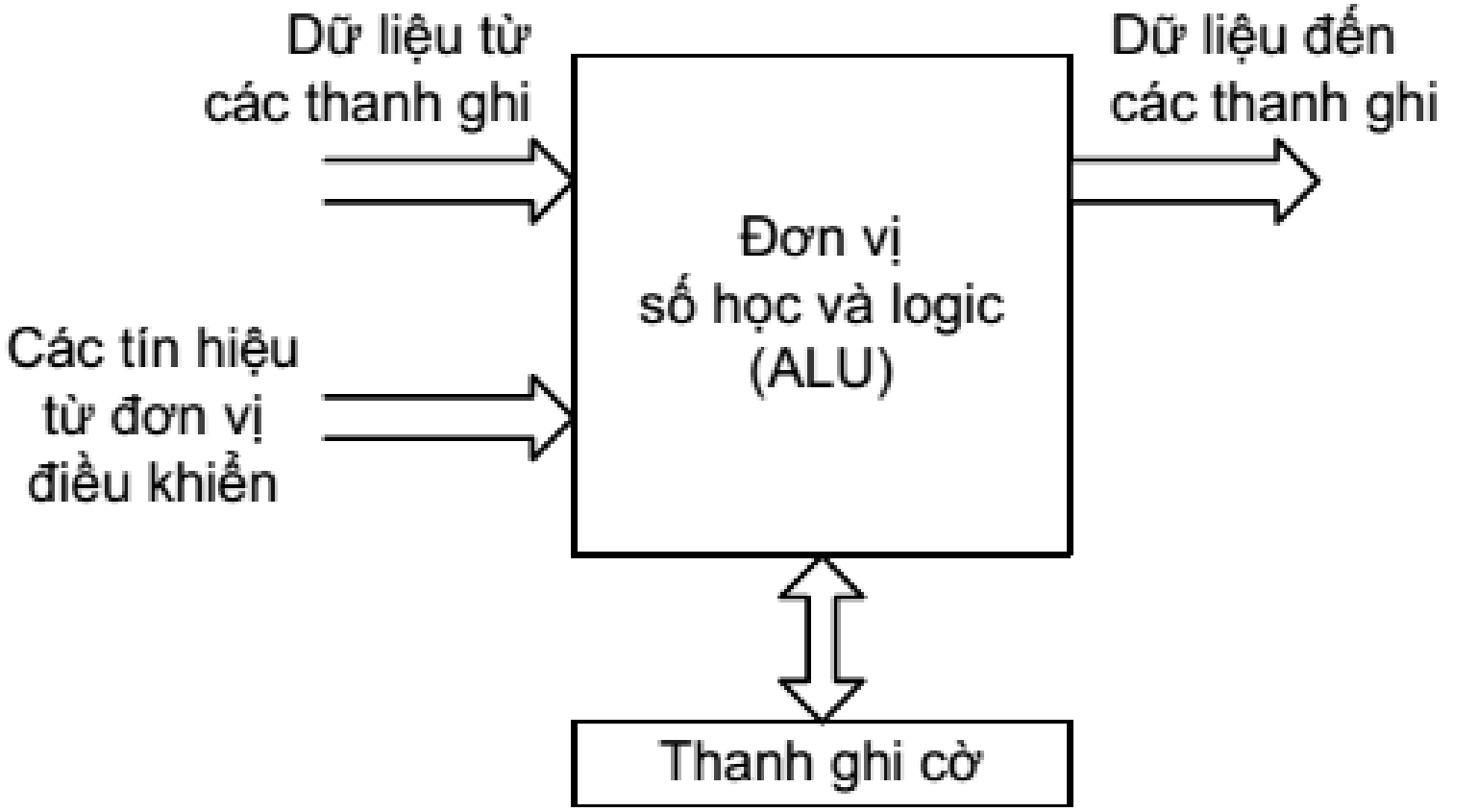


1. Tổ chức của CPU

- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Mô hình kết nối ALU

- Thực hiện các phép toán số học và phép toán logic:
  - Số học: cộng, trừ, nhân, chia, đảo dấu
  - Logic: AND, OR, XOR, NOT, phép dịch bit



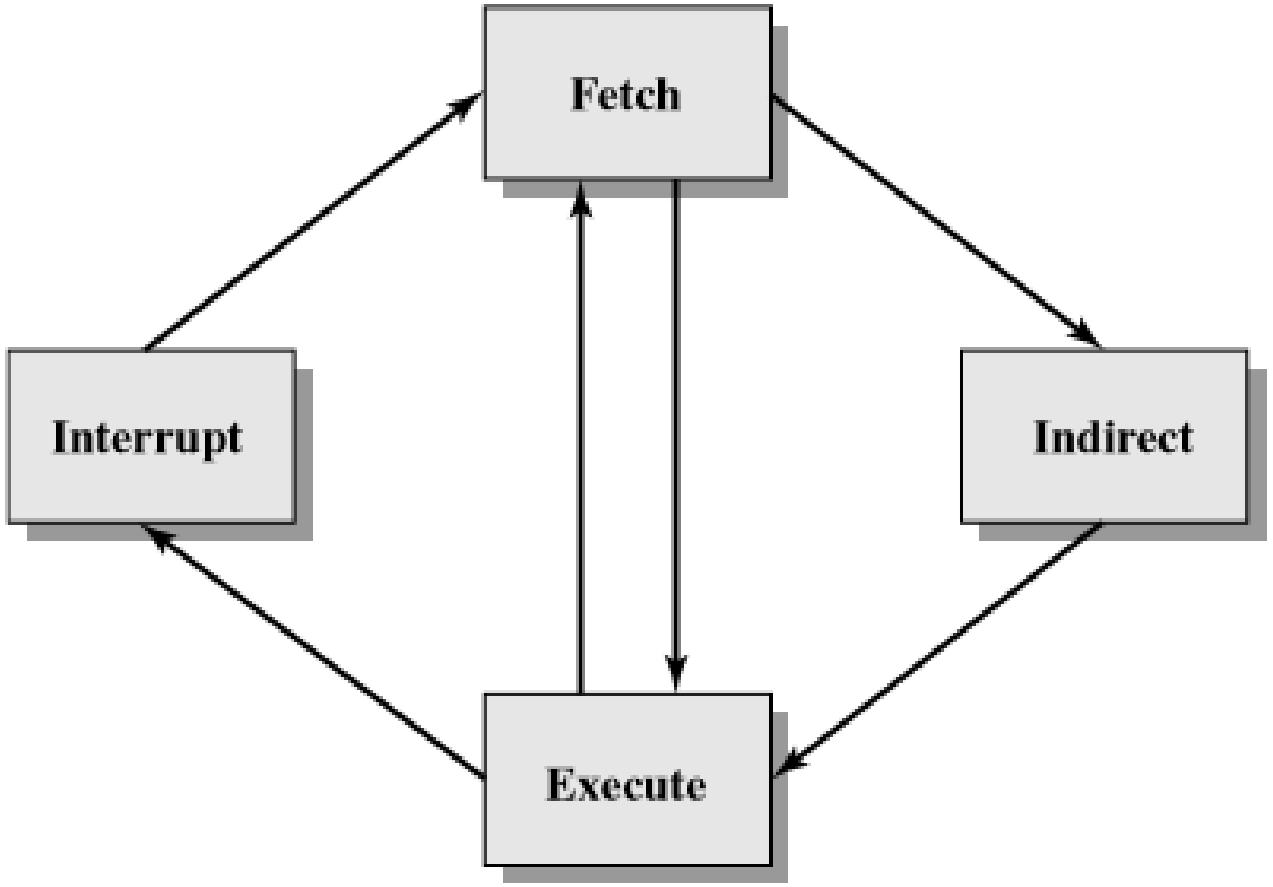
1. Tổ chức của CPU

- Nhiệm vụ
- Sơ đồ cấu trúc
- Thanh ghi
- Đơn vị điều khiển (CU)
- Đơn vị số học và logic (ALU)
- Chu trình lệnh

Chu trình lệnh (Instruction Cycle)

- Nhận lệnh
- Giải mã lệnh
- Nhận toán hạng
- Thực hiện lệnh
- Cất toán hạng
- Ngắt

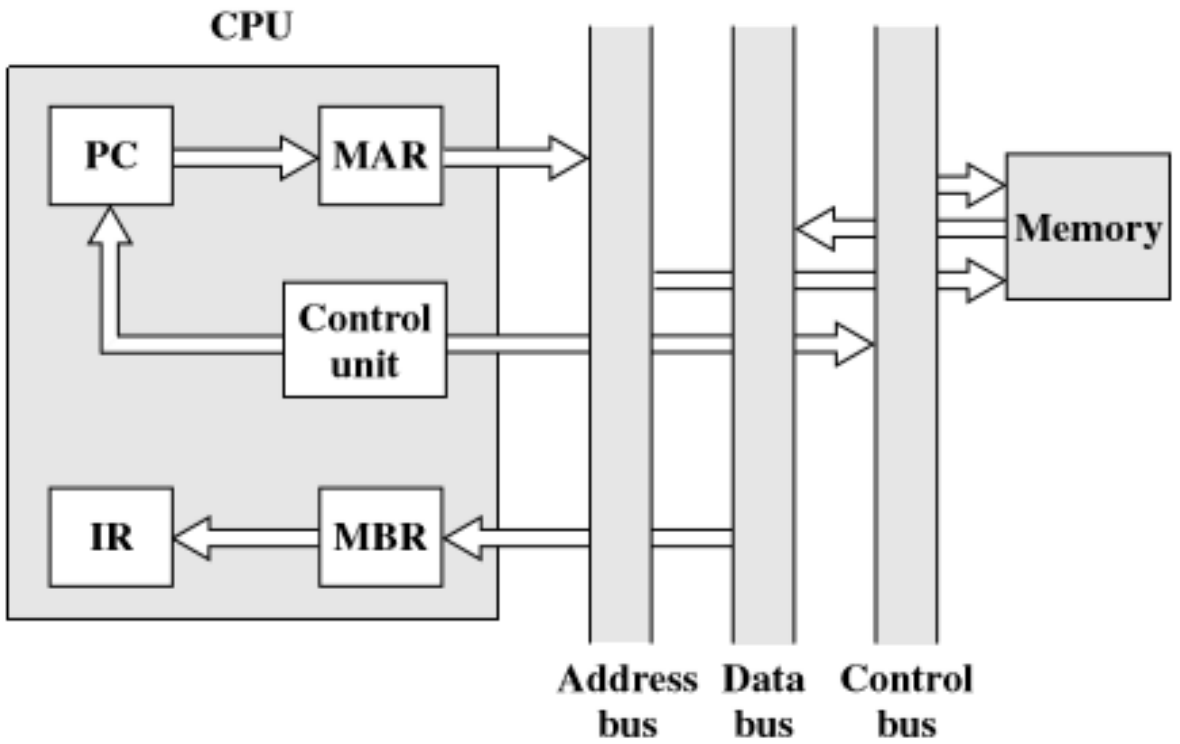
Giản đồ trạng thái của chu trình lệnh



The Instruction Cycle

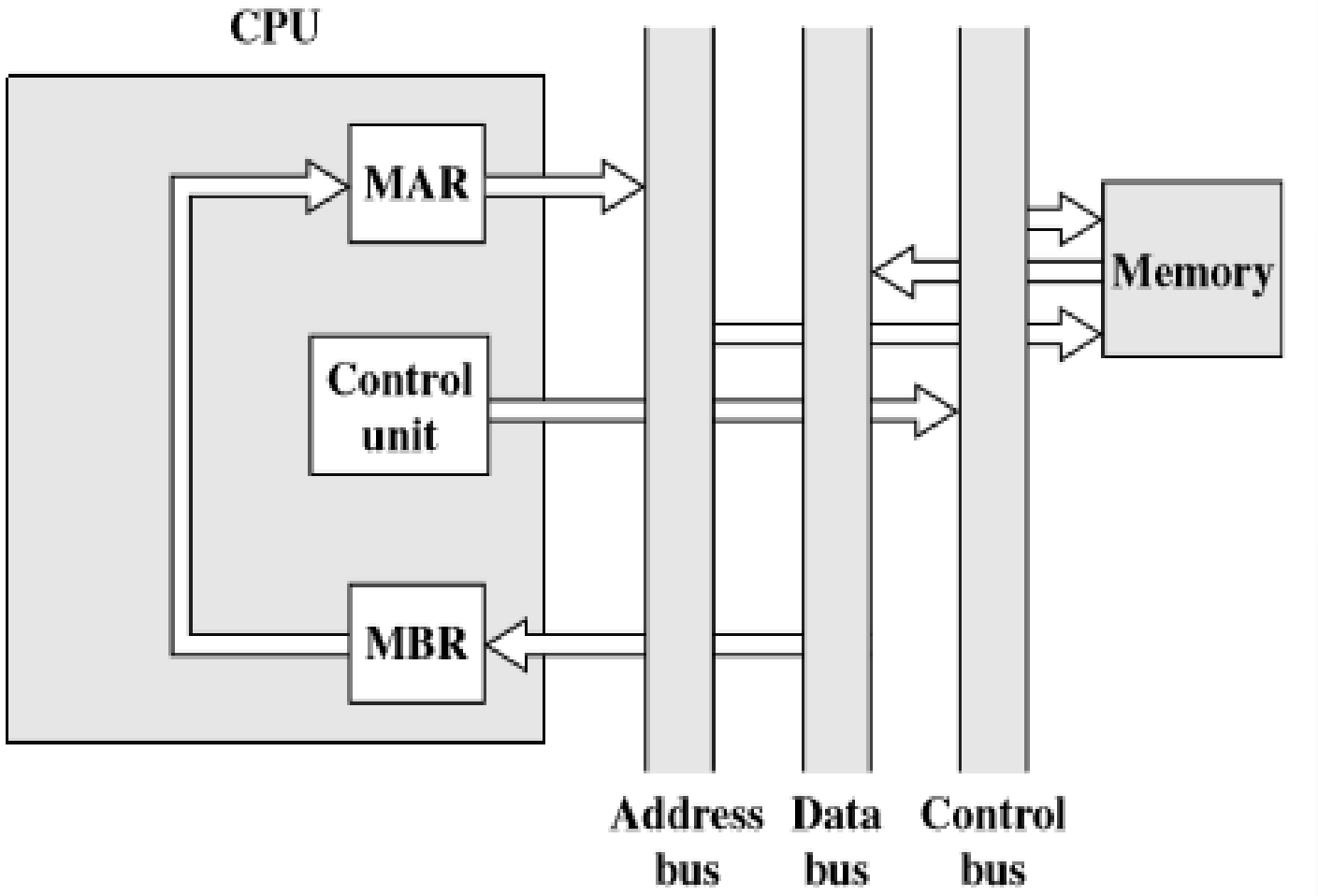
Fetch Cycle

- Địa chỉ của lệnh trong PC được chuyển vào MAR và đặt lên bus địa chỉ
- CPU phát tín hiệu yêu cầu đọc bộ nhớ
- Lệnh từ bộ nhớ được đặt lên bus dữ liệu và được copy vào MBR rồi chuyển vào IR
- CPU tăng nội dung PC để trỏ sang lệnh kế tiếp



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

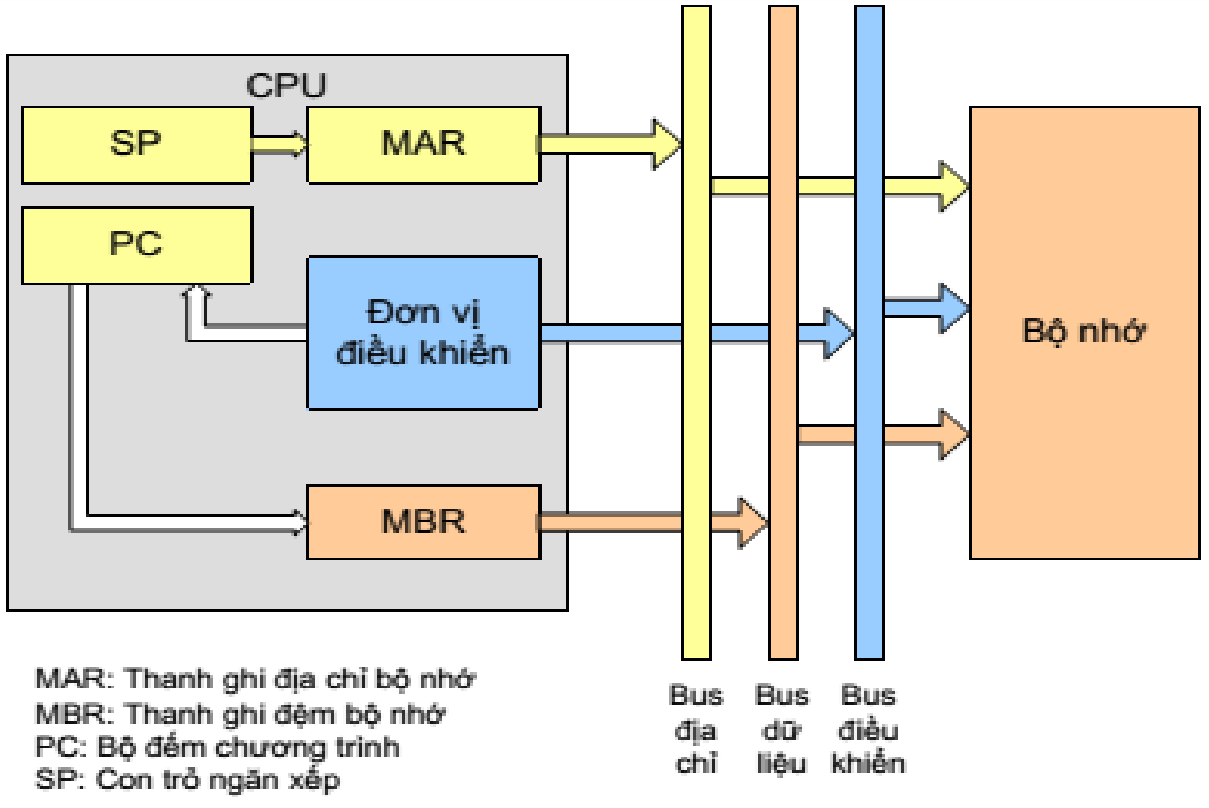
Indirect Cycle



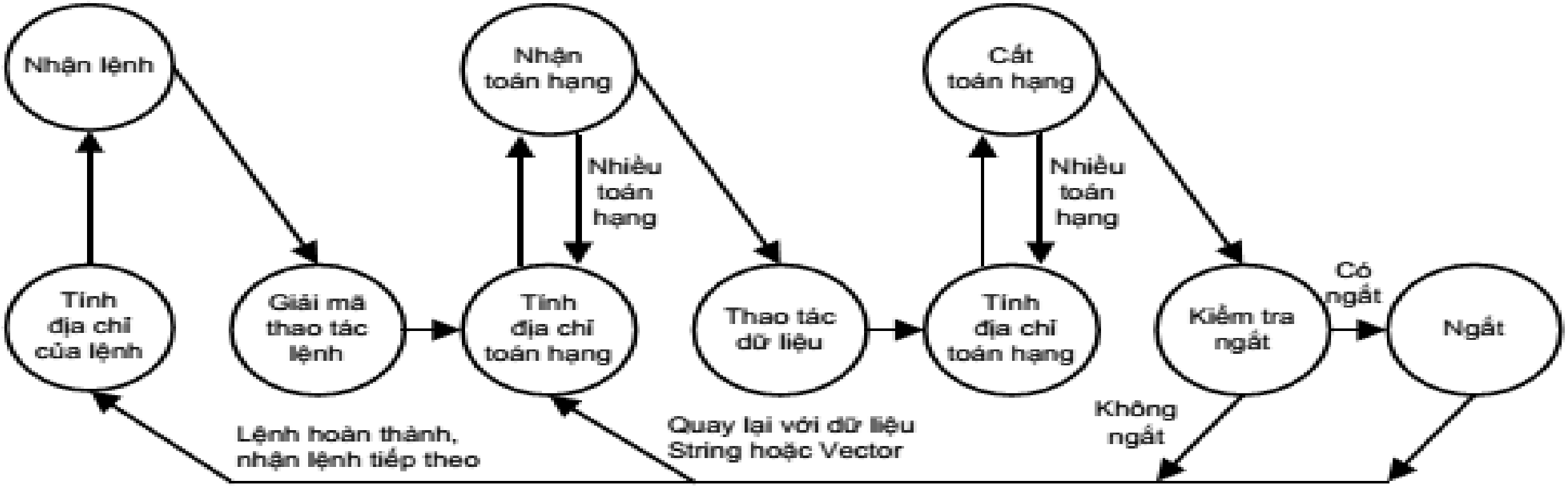
Data Flow, Indirect Cycle

Interrupt Cycle

- Nội dung của PC (địa chỉ trở về sau khi ngắt) được đưa ra bus dữ liệu
- CPU đưa địa chỉ (thường là con trỏ ngăn xếp SP) ra bus địa chỉ
- CPU phát tín hiệu yêu cầu ghi bộ nhớ
- Địa chỉ lệnh đầu tiên của chương trình con xử lý ngắt được nạp vào PC



Giản đồ trạng thái của chu trình lệnh

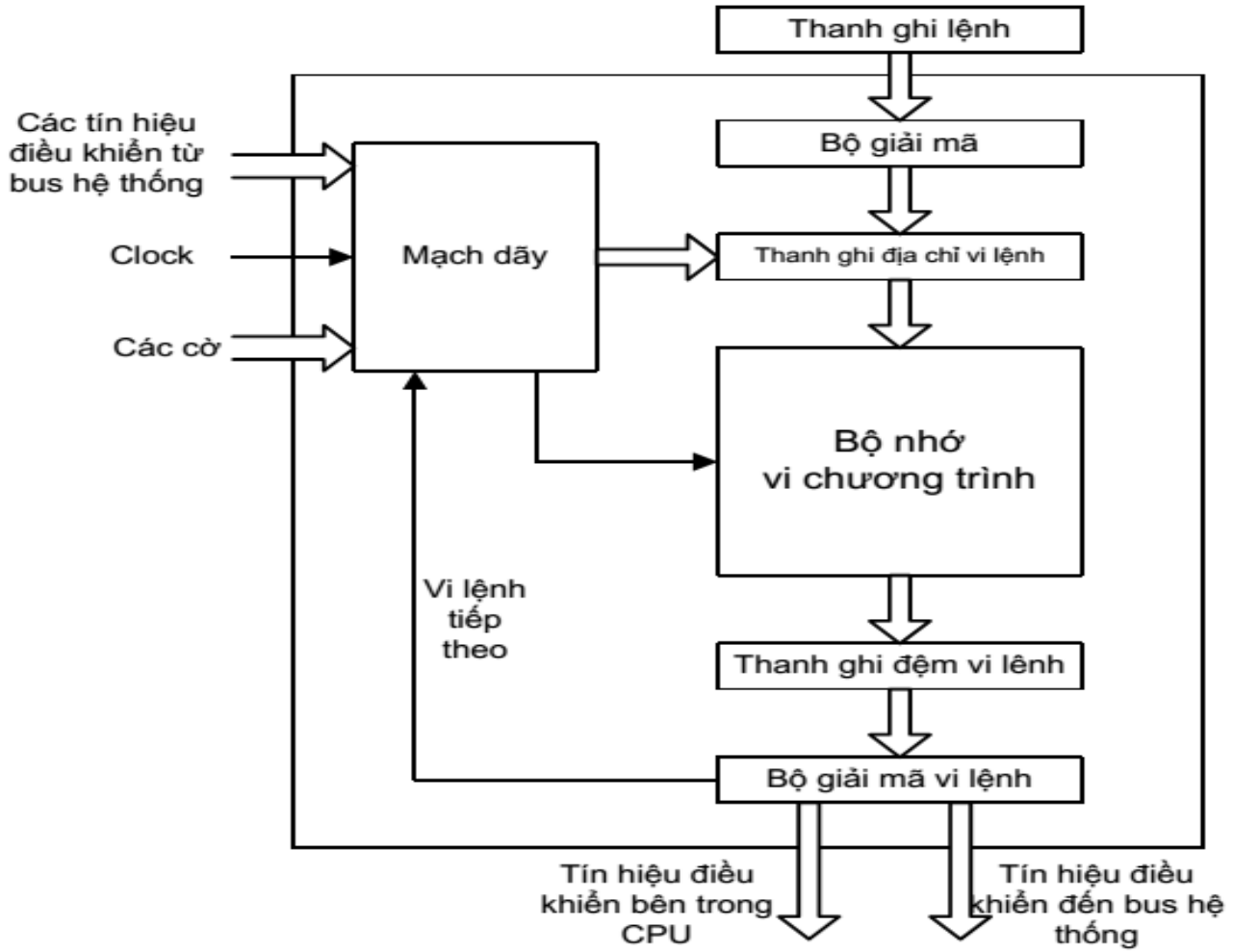


2. Các phương pháp thiết kế đơn vị điều khiển

- Đơn vị điều khiển vi chương trình
- Đơn vị điều khiển kết nối phần cứng

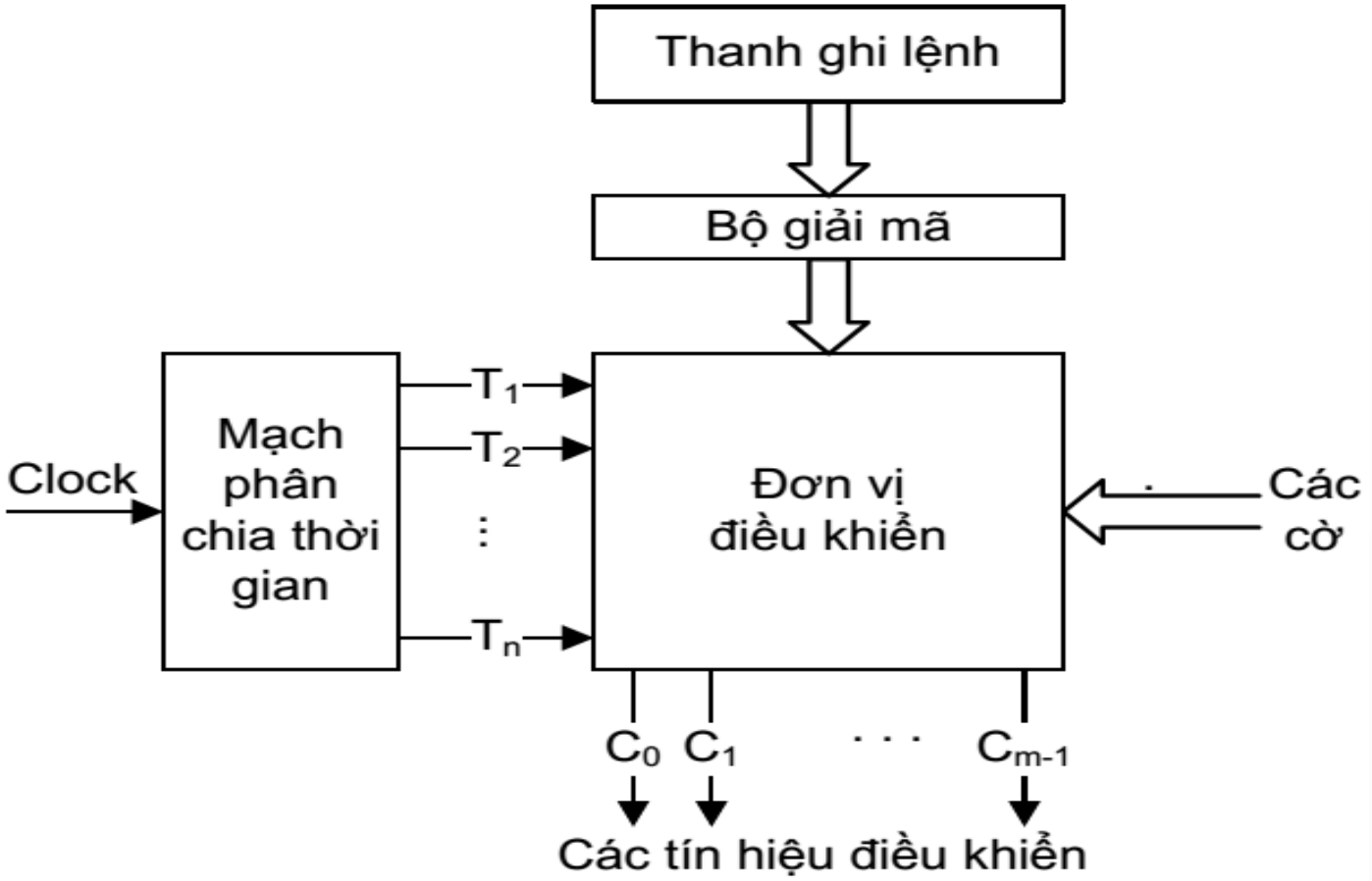
Đơn vị điều khiển vi chương trình

- Bộ nhớ ROM lưu trữ các vi chương trình (micro program)
- Một vi chương trình bao gồm các vi lệnh (micro instruction)
- Mỗi vi lệnh mã hoá cho một vi thao tác (micro operation)
- Để hoàn thành một lệnh cần thực hiện một hoặc một vài vi chương trình
- Tốc độ chậm



Đơn vị điều khiển nối kết cứng

- Sử dụng mạch thiết kế cứng để giải mã và tạo các tín hiệu điều khiển thực hiện lệnh
- Tốc độ nhanh
- Đơn vị điều khiển phức tạp



Chương 6

1. Tổ chức của CPU
2. Thiết kế đơn vị điều khiển
3. Kỹ thuật đường ống lệnh

3. Kỹ thuật đường ống lệnh (Instruction Pipelining)

- Chia chu trình lệnh thành nhiều công đoạn, thực hiện gối lên nhau (như dây chuyền lắp ráp)
- Ví dụ, bộ xử lý MIPS có 6 công đoạn:
  - FI (Fetch Instruction): Nhận lệnh từ bộ nhớ
  - DI (Decode Instruction): Giải mã lệnh và đọc thanh ghi
  - CO (Calculate Operand): Tính toán địa chỉ toán hạng
  - FO (Fetch Operand): Nạp toán hạng
  - EI (Execute Instruction): Thực hiện lệnh
  - WO (Write Operand): Ghi kết quả

Biểu đồ thời gian của đường ống lệnh

Time →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

- Thời gian thực hiện 1 công đoạn = T
- Thời gian thực hiện tuần tự 9 lệnh = 54T
- Thời gian thực hiện đường ống 9 lệnh = 14T

Các trở ngại (Hazard) của kỹ thuật đường ống lệnh

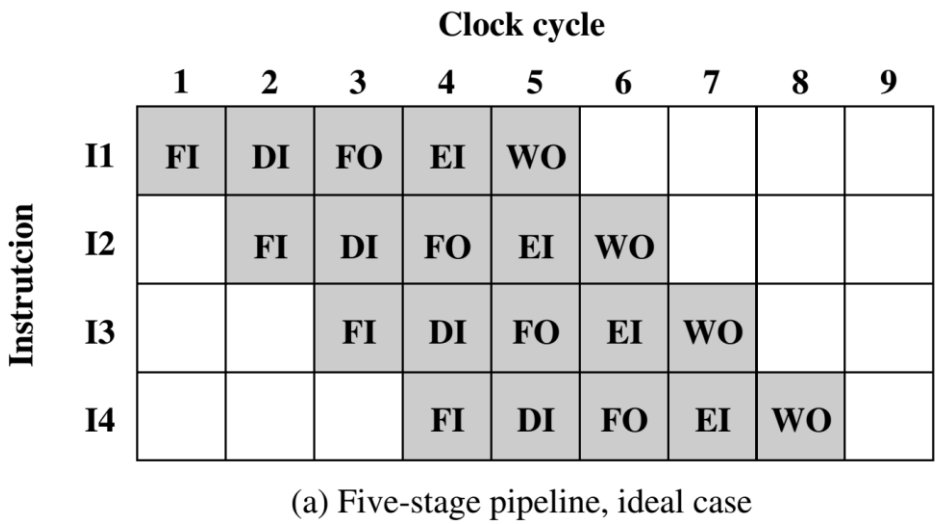
- Hazard: Tình huống ngăn cản bắt đầu của lệnh tiếp theo ở chu kỳ tiếp theo
  - Hazard tài nguyên: Do tài nguyên được yêu cầu đang bận
  - Hazard dữ liệu: Cần phải đợi để lệnh trước hoàn thành việc đọc/ghi dữ liệu
  - Hazard điều khiển: Do rẽ nhánh gây ra

Hazard tài nguyên

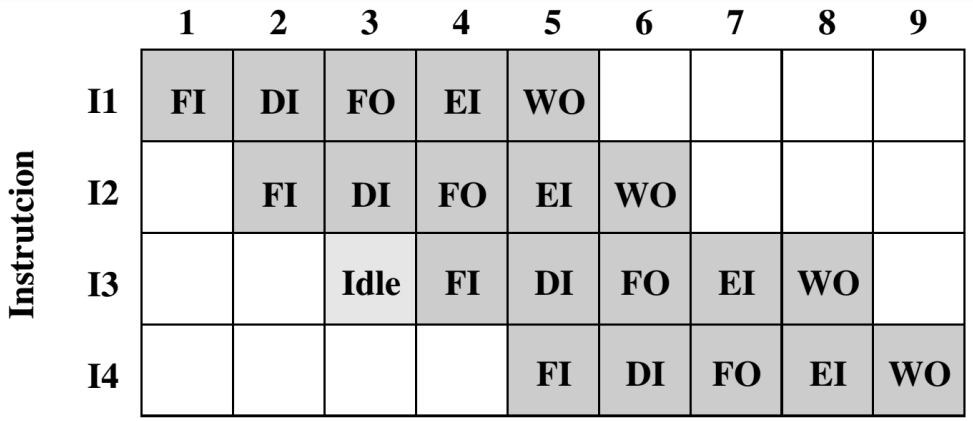
- Xung đột khi sử dụng chung tài nguyên
- Giả sử chỉ có một cổng bộ nhớ dùng chung
  - Lệnh Load/store yêu cầu truy cập dữ liệu
  - Giả sử: I1 đọc DL từ memory, các toán hạng khác trong thanh ghi, thì I3 trễ 1 chu kỳ

=>Datapath cho bộ nhớ lệnh và bộ nhớ dữ liệu tách rời (hoặc cache lệnh/cache dữ liệu tách rời)

  - *VD: nhiều chỉ thị cùng đòi hỏi ALU*
- *Tăng tài nguyên dùng chung*

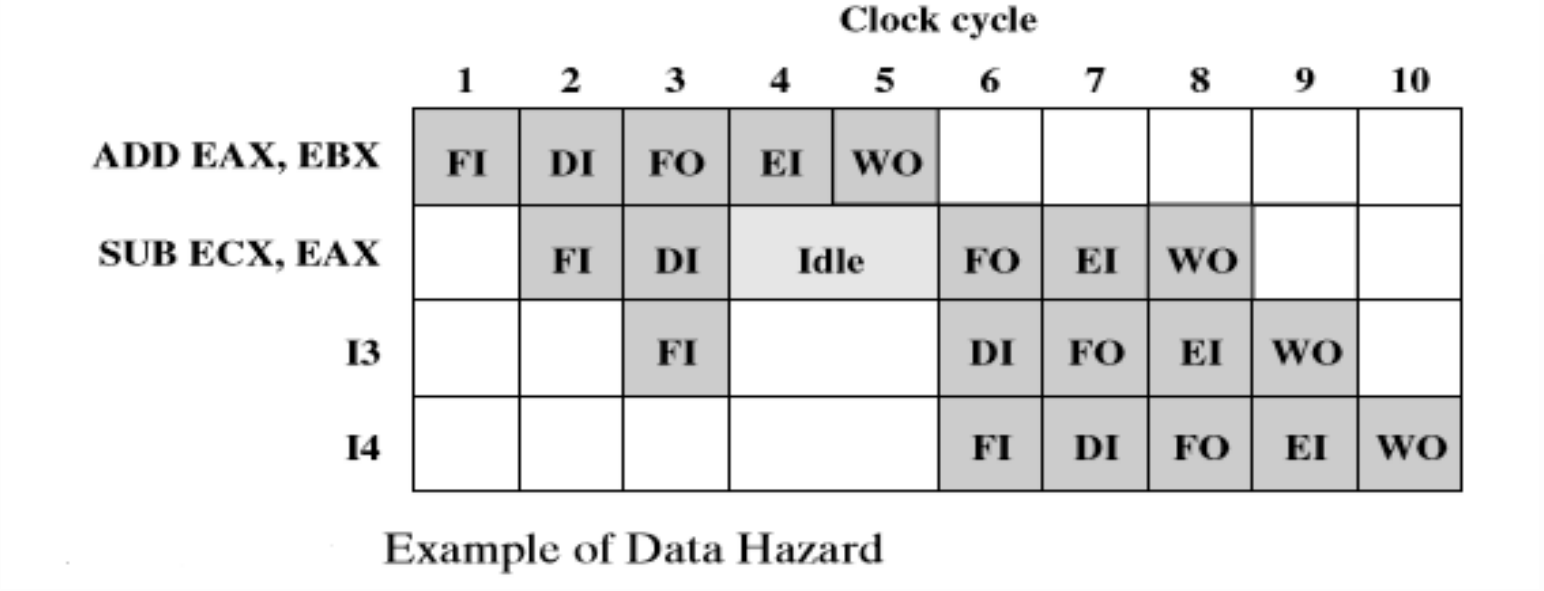


(a) Five-stage pipeline, ideal case



(b) I1 source operand in memory

Hazard dữ liệu

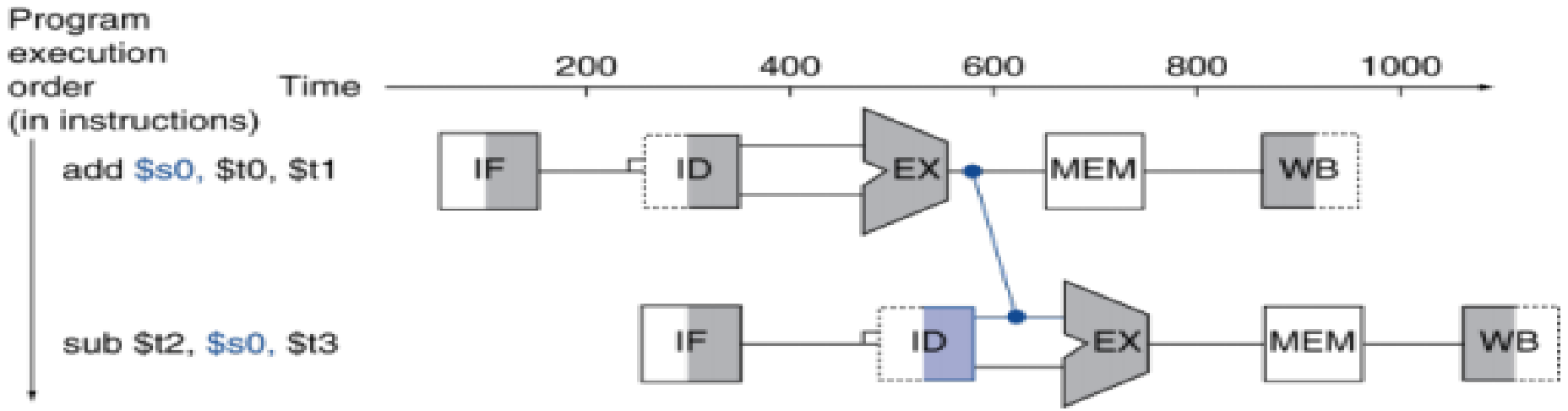


Example of Data Hazard

- Lệnh phụ thuộc vào việc hoàn thành truy cập dữ liệu của lệnh trước đó.
- Giải pháp:
  - Đổi thứ tự thực hiện lệnh
  - Hoặc thêm phần cứng để gửi vượt trước (forwarding)

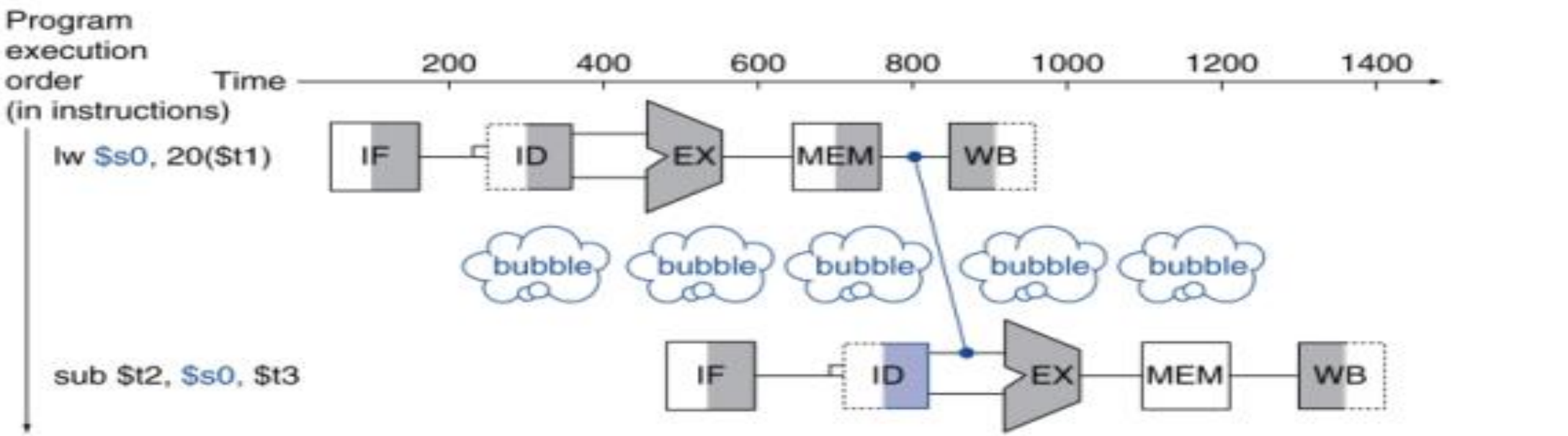
Hazard dữ liệu

- Forwarding
  - Sử dụng kết quả ngay sau khi nó được tính
  - Yêu cầu có đường kết nối thêm trong datapath



Hazard dữ liệu

- Forwarding
  - Một số trường hợp vẫn phải chèn trì hoãn khi không giải quyết hoàn toàn bằng forwarding

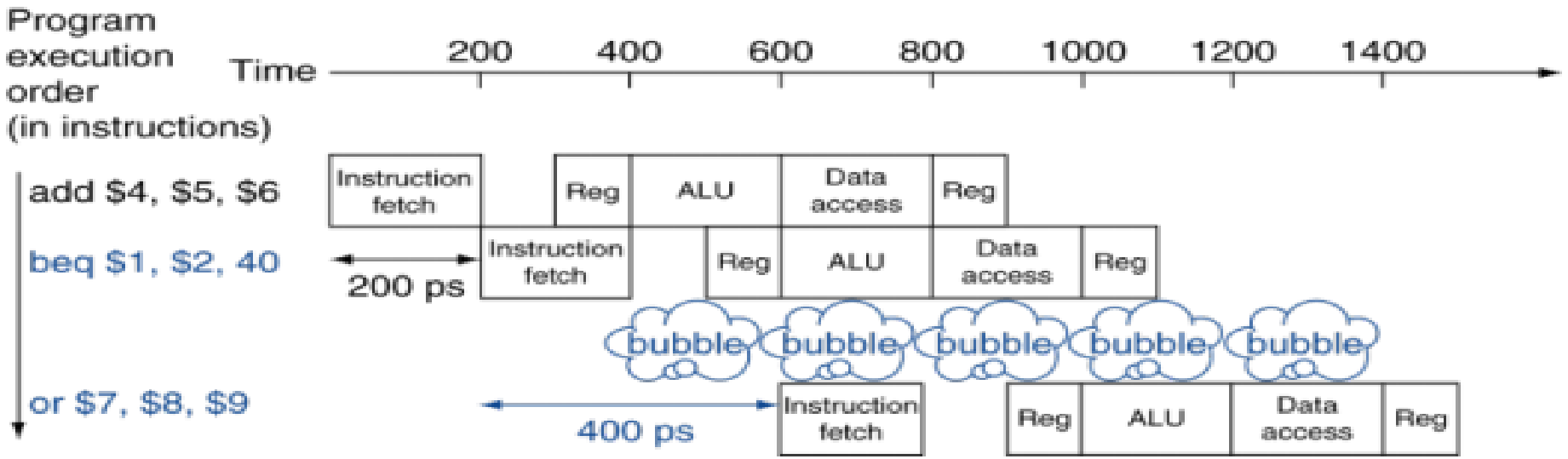


Hazard điều khiển

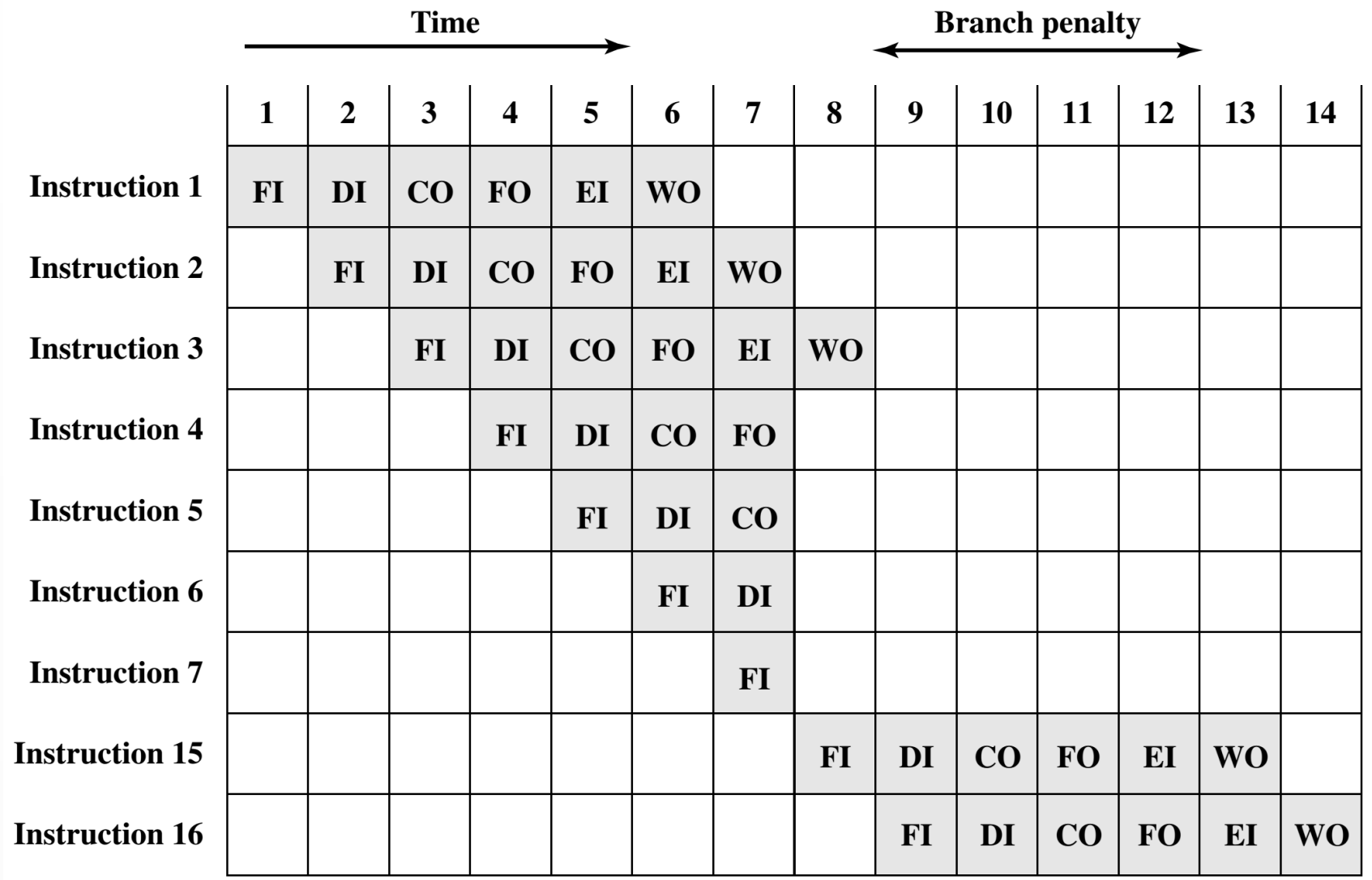
- Rẽ nhánh
  - Lệnh tiếp theo phụ thuộc vào kết quả rẽ nhánh
  - Đường ống không thể luôn nhận đúng lệnh
    - Lệnh rẽ nhánh đang ở công đoạn giải mã lệnh (ID)
- Cách khắc phục:
  - Trì hoãn khi rẽ nhánh, nếu đường ống dài thì ko hiệu quả.
  - Dự đoán rẽ nhánh, chỉ trì hoãn khi dự đoán là sai
- Với MIPS
  - Có thể dự đoán rẽ nhánh không xẩy ra
  - Nhận lệnh ngay sau lệnh rẽ nhánh (không làm trễ)

Trì hoãn khi rẽ nhánh

- Đợi cho đến khi kết quả rẽ nhánh đã được xác định trước khi nhận lệnh tiếp theo



MIPS với dự đoán rẽ nhánh không xảy ra



Tóm lại

- Kỹ thuật đường ống cải thiện hiệu năng bằng cách tăng số lệnh thực hiện tại một thời điểm
  - Thực hiện nhiều lệnh đồng thời
  - Mỗi lệnh có cùng thời gian thực hiện
- Các dạng hazard: 3 dạng
  - Tài nguyên, dữ liệu, điều khiển
- Thiết kế tập lệnh ảnh hưởng đến độ phức tạp của việc thực hiện đường ống

HẾT CHƯƠNG 6