

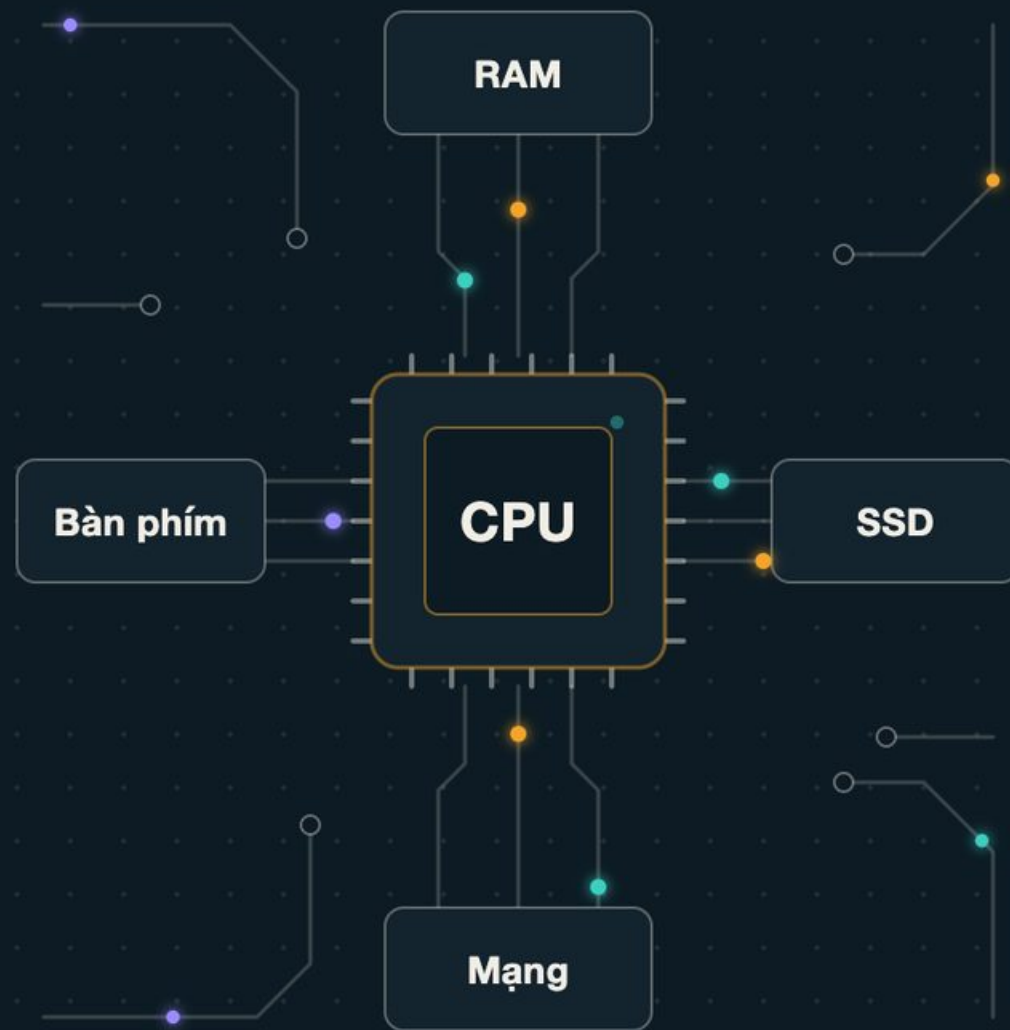
CEA201 · CHỦ ĐỀ 3 · CHƯƠNG 8

Vào / Ra

Input / Output

Thiết bị ngoại vi · Ngắt · DMA · NVMe

Nhóm 4 · Fall 2026



Mục lục

Từ thiết bị ngoại vi đến SSD NVMe trong máy tính hiện đại

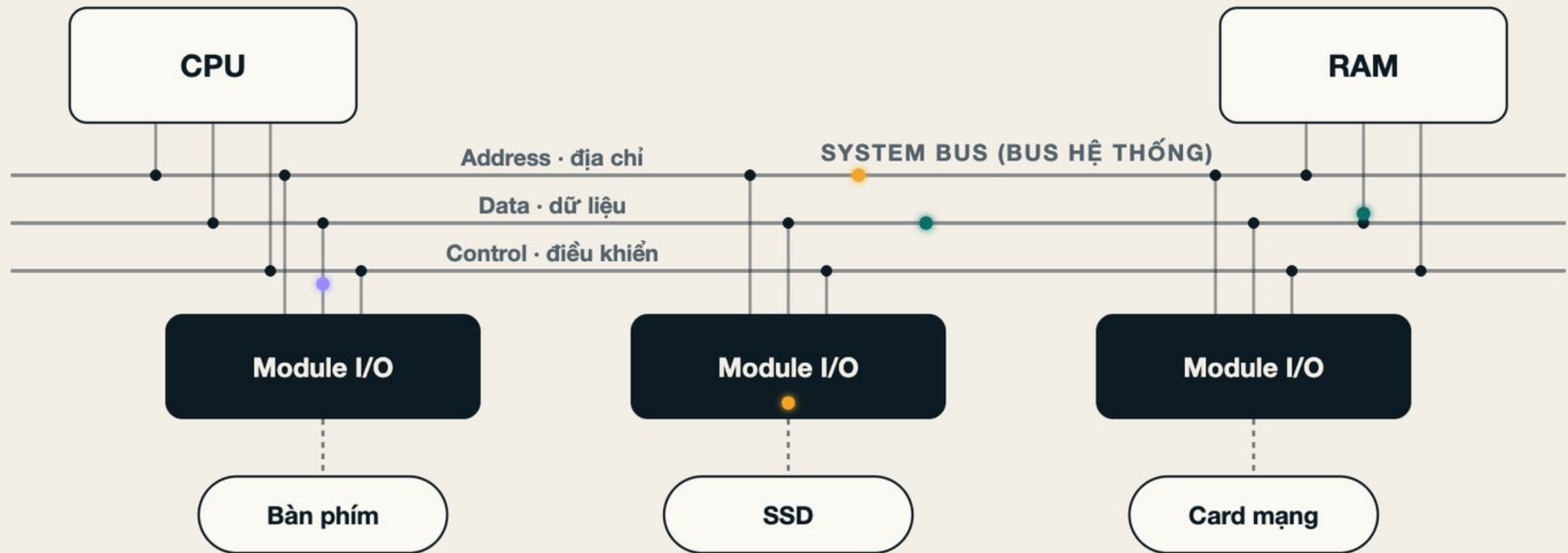
MÃ MÀU XUYỀN SUỐT

- **Polling** · CPU hỏi liên tục
- **Ngắt** · thiết bị báo khi xong
- **DMA** · dữ liệu vào thẳng RAM

●	01 · 8.1	Thiết bị ngoại vi	External devices
●	02 · 8.2	Module I/O	I/O modules
●	03 · 8.3	I/O lập trình	Programmed I/O
●	04 · 8.4	I/O điều khiển bằng ngắt	Interrupt-driven I/O
●	05 · 8.5	Truy cập bộ nhớ trực tiếp	DMA
●	06 · 8.6	Kênh và bộ xử lý I/O	I/O channels
●	07	NVMe SSD trên PCIe	Liên hệ hiện đại
●	08	So sánh và tổng kết	Comparison

MỞ ĐẦU

Bức tranh tổng thể



CPU không nói chuyện trực tiếp với thiết bị: mọi trao đổi đều đi qua **module I/O**.

01

PHẦN 01 · MỤC 8.1

Thiết bị ngoại vi

External Devices · cầu nối giữa máy tính và thế giới bên ngoài

Ba nhóm thiết bị ngoại vi



HUMAN READABLE

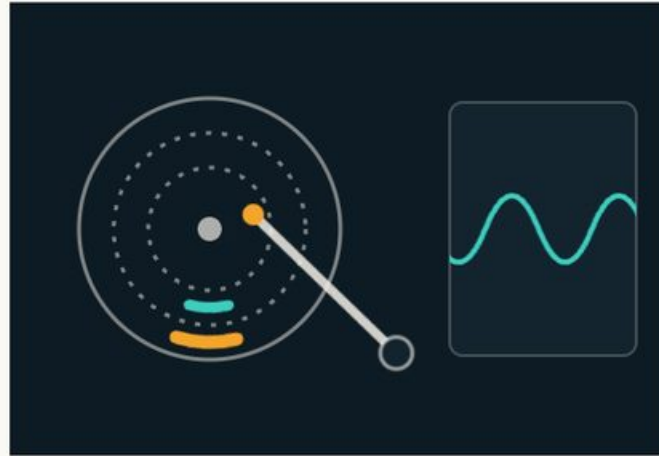
Người đọc được

Giao tiếp với **người dùng**

Màn hình

Máy in

Bàn phím



MACHINE READABLE

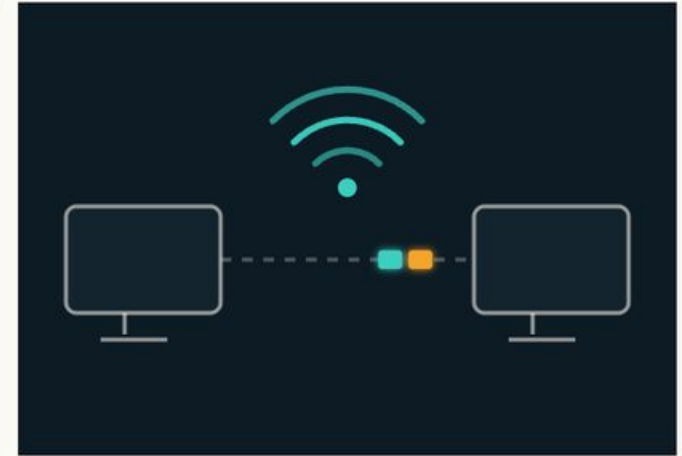
Máy đọc được

Giao tiếp với **thiết bị khác**

Ổ đĩa

Cảm biến

Bộ chấp hành



COMMUNICATION

Truyền thông

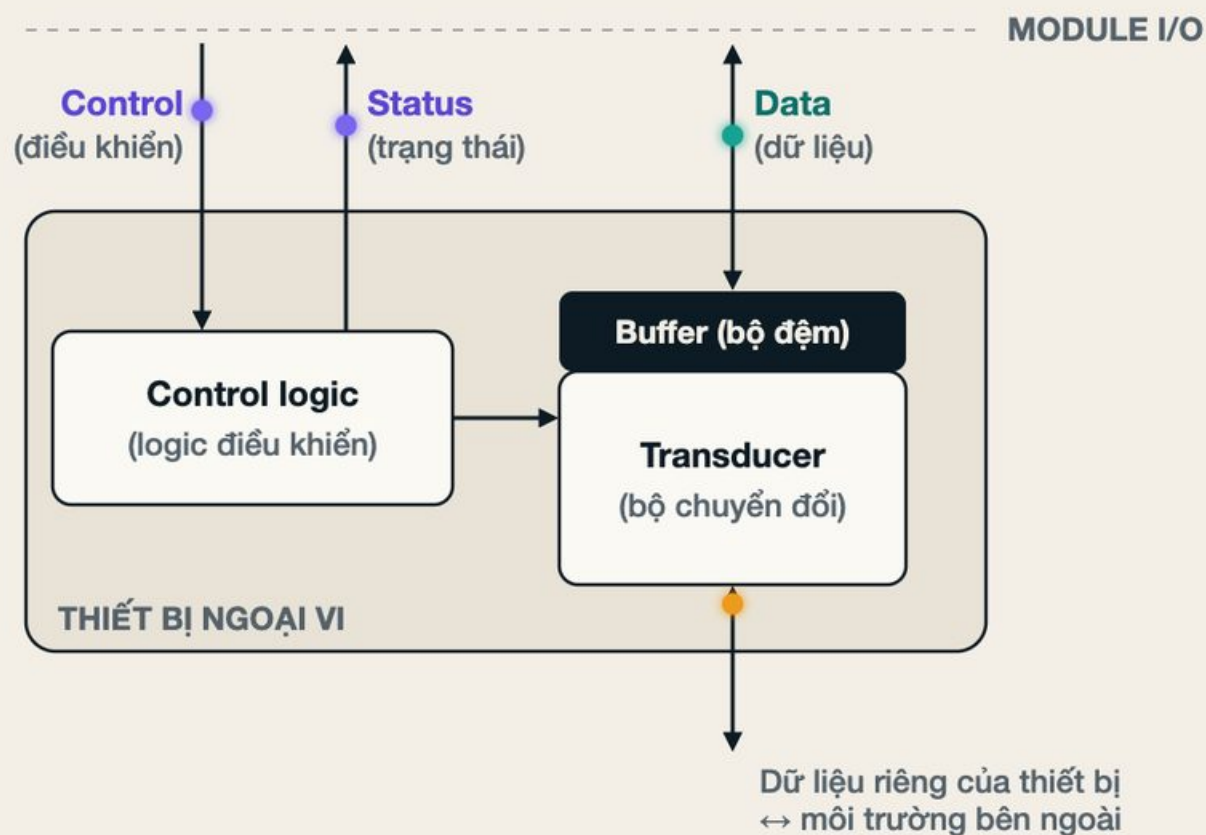
Giao tiếp với **thiết bị ở xa**

Modem

Card mạng

Máy tính khác

Bên trong một thiết bị ngoại vi



Control logic (logic điều khiển)

Nhận lệnh từ module I/O, điều khiển thiết bị, báo lại trạng thái

Transducer (bộ chuyển đổi)

Đổi qua lại giữa tín hiệu điện và dạng khác: ánh sáng, âm thanh, lực nhấn

Buffer (bộ đệm)

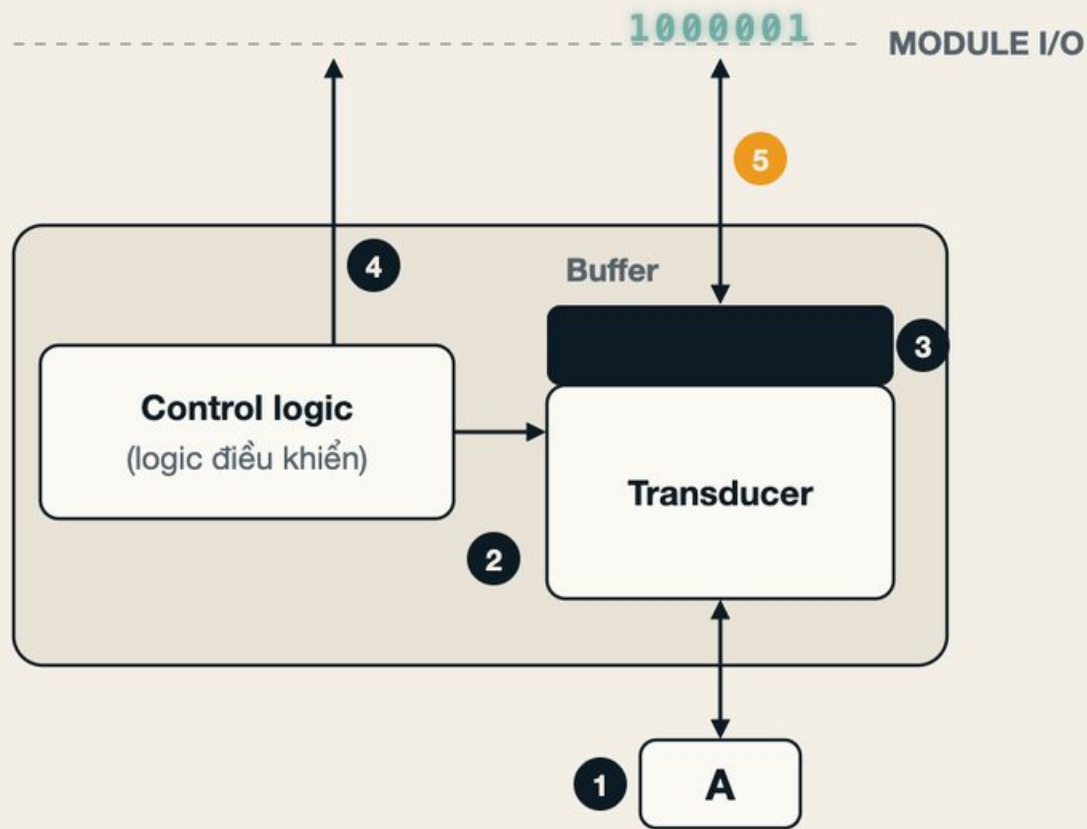
Giữ tạm dữ liệu đang trao đổi với module I/O

3 loại tín hiệu với module I/O

Control ↓ lệnh · Status ↑ bận, sẵn sàng, lỗi

Data ↑ dữ liệu đi hai chiều

Ví dụ: gõ phím 'A' trên bàn phím



- 1 Phím 'A' được nhấn
- 2 **Transducer** đổi lực nhấn thành tín hiệu điện
- 3 Tín hiệu thành mã IRA 7 bit **1000001**, nằm trong **buffer**
- 4 **Control logic** báo trạng thái “sẵn sàng” cho module I/O
- 5 Mẫu bit được gửi lên module I/O qua đường **data**

IRA (International Reference Alphabet) = mã ASCII
7 bit → **128 ký tự**: ký tự in được và ký tự điều khiển

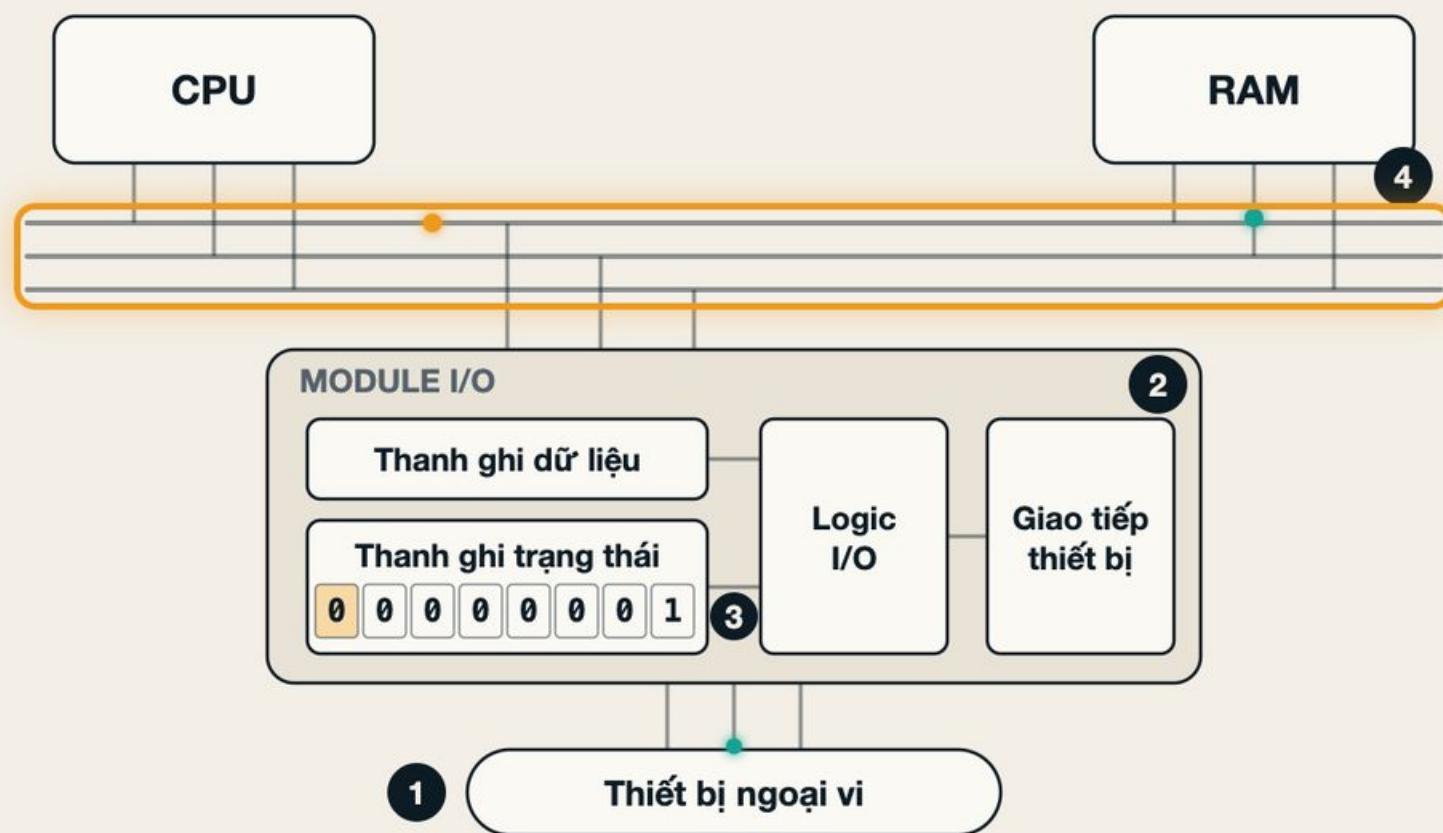
02

PHẦN 02 · MỤC 8.2

Module I/O

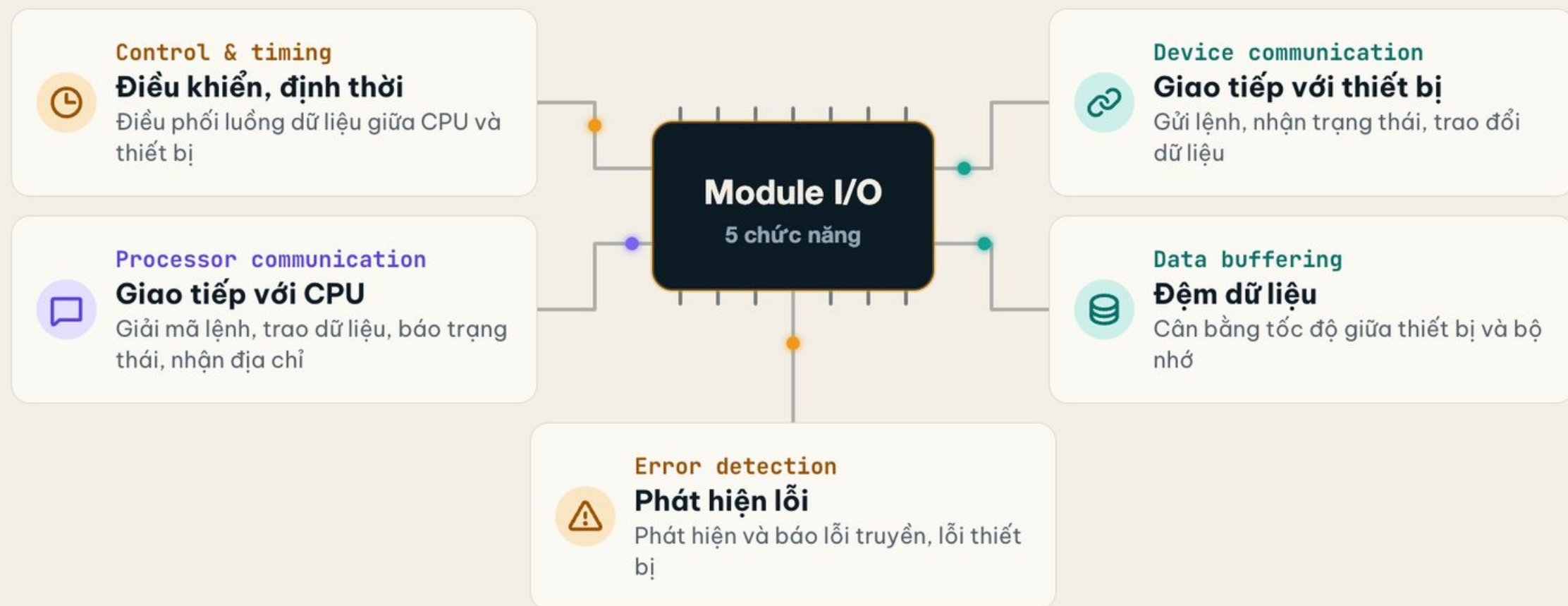
I/O Modules · người phiên dịch giữa CPU và thiết bị

Bốn thành phần cần nhận diện



- 1 Peripheral (thiết bị ngoại vi)**
Nơi dữ liệu sinh ra hoặc đi đến: bàn phím, SSD, màn hình
- 2 I/O controller (bộ điều khiển I/O)**
Chính là module I/O: nhận lệnh, đệm dữ liệu, điều khiển thiết bị
- 3 Status register (thanh ghi trạng thái)**
Các bit cờ READY, BUSY, ERROR để CPU đọc
- 4 System bus (bus hệ thống)**
Đường địa chỉ, dữ liệu, điều khiển nối CPU, RAM và module

Năm chức năng của module I/O

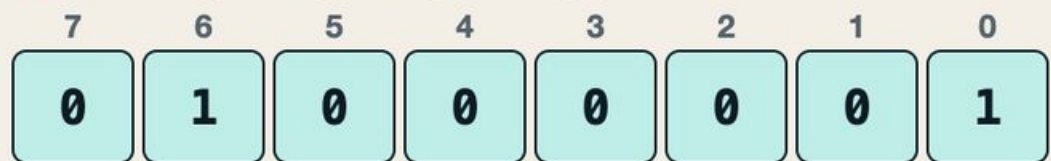


Status register (thanh ghi trạng thái)

517 Status / control register



516 Data register (thanh ghi dữ liệu)



CPU

Bước 4 • Đọc ô 516

nhận 01000001 = 'A'

VÍ DỤ: CPU ĐỌC 1 KÝ TỰ TỪ BÀN PHÍM

- 1 CPU ghi 1 vào bit 0 của ô 517 → ra lệnh bắt đầu đọc
- 2 CPU đọc lại ô 517: bit 7 = 0 nghĩa là còn bận → đọc tiếp. Đây chính là **polling** (hỏi vòng)
- 3 Bit 7 = 1 → ký tự đã nằm sẵn trong ô 516
- 4 CPU đọc ô 516 → nhận **01000001** = 'A'

CPU biết thiết bị bận hay xong **chỉ bằng cách đọc thanh ghi trạng thái.**

Bốn lệnh I/O (I/O commands)

Khi CPU gọi đúng địa chỉ của một module I/O, nó gửi một trong bốn lệnh:

CONTROL

Điều khiển

CPU → module → thiết bị

Kích hoạt thiết bị và cho biết phải làm gì

Ví dụ: bảo ổ băng từ tua lại, hoặc tiến tới bản ghi kế tiếp

TEST

Kiểm tra

CPU ⇌ module: trạng thái?

Hỏi các điều kiện trạng thái của module và thiết bị

Ví dụ: thiết bị đã bật chưa? Lệnh trước xong chưa? Có lỗi không?

READ

Đọc

thiết bị → buffer của module

Module lấy dữ liệu từ thiết bị vào buffer; CPU lấy qua bus dữ liệu

Ví dụ: đọc ký tự vừa gõ từ bàn phím

WRITE

Ghi

bus dữ liệu → module → thiết bị

Module lấy dữ liệu từ bus dữ liệu rồi gửi ra thiết bị

Ví dụ: gửi một ký tự ra máy in

03

PHẦN 03 · MỤC 8.3

I/O lập trình

Programmed I/O · CPU tự ra lệnh, tự chờ, tự chuyển dữ liệu

Ba kỹ thuật vào/ra

Table 8.1

KHÔNG DỪNG NGẮT

CÓ DỪNG NGẮT (INTERRUPT)

Đi qua CPU

I/O → CPU → RAM



Programmed I/O
(I/O lập trình)

“Cứ vài phút lại ra cửa xem shipper tới chưa.”



Interrupt-driven I/O
(I/O điều khiển bằng ngắt)

“Ngồi làm việc, chuông cửa reo thì ra nhận hàng.”

Đi thẳng

I/O → RAM

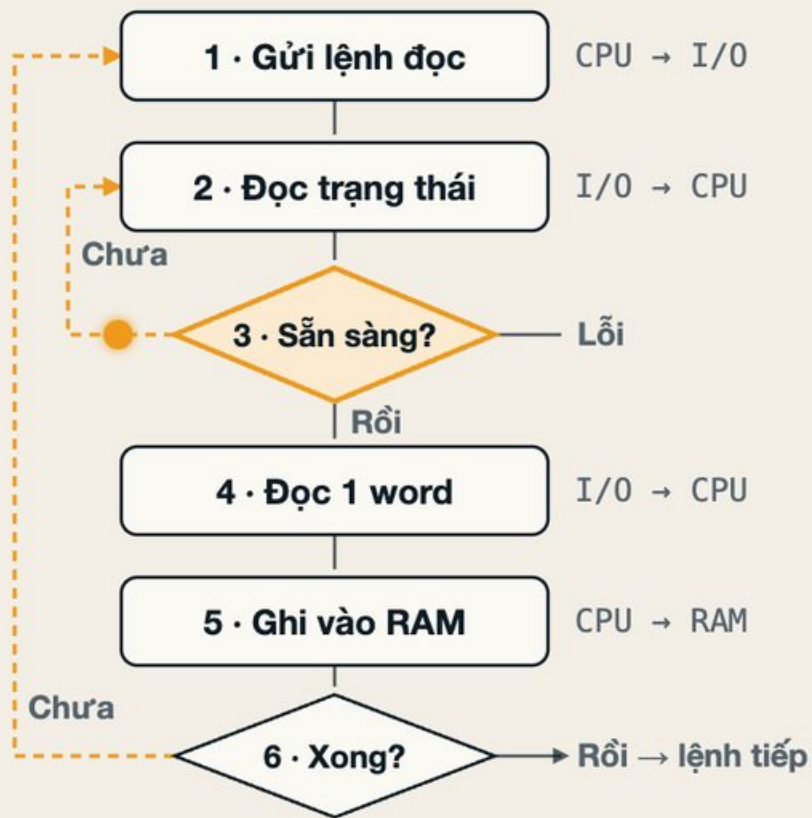
(không có kỹ thuật này)



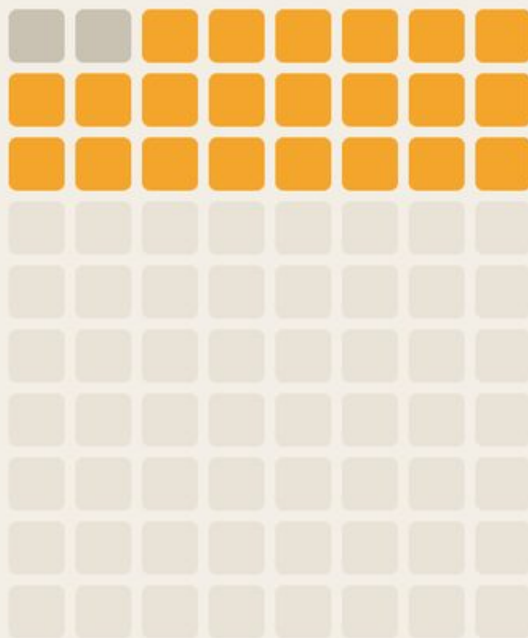
DMA
(Direct Memory Access: truy cập bộ nhớ trực tiếp)

“Shipper có chìa khoá: tự cất hàng vào kho, xong thì nhắn tin.”

Programmed I/O (I/O lập trình): các bước



CPU đang làm gì?



Orange chờ: hỏi trạng thái
 Teal chuyển dữ liệu

AI KHỞI TẠO?

CPU

Gửi lệnh đọc tới module I/O

AI CHỜ?

CPU

Busy waiting (chờ bận): lặp đọc trạng thái liên tục

AI CHUYỂN DỮ LIỆU?

CPU

Từng word: I/O → CPU → RAM

Hai cách đánh địa chỉ thiết bị

Memory-mapped I/O (ánh xạ bộ nhớ)



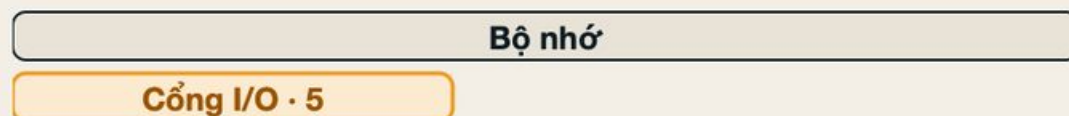
Một không gian địa chỉ chung cho RAM và I/O

```

200  LOAD AC      "1"    ; nạp 1
      STORE AC     517    ; bắt đầu đọc
▶ 202  LOAD AC     517    ; đọc trạng thái
      BRANCH Sign=0 202   ; chưa xong → lặp
      LOAD AC      516    ; lấy dữ liệu
  
```

Dùng chung lệnh đọc/ghi bộ nhớ: **nhiều lệnh** để chọn

Isolated I/O (I/O tách biệt)



Hai không gian địa chỉ tách riêng

```

200  LOAD I/O     5      ; bắt đầu đọc
201  TEST I/O     5      ; kiểm tra
▶    BRANCH Not Ready 201 ; chưa xong → lặp
      IN          5      ; lấy dữ liệu
  
```

Cần **lệnh I/O riêng** (tập lệnh hạn chế) và đường chọn I/O hay bộ nhớ

Q1

Khi nào polling (hỏi vòng) lãng phí, và khi nào vẫn hợp lý?

Thiết bị chậm

vd: bàn phím, máy in

hỏi... chưa · hỏi... chưa · hỏi... chưa · ...

→ gần như toàn bộ thời gian CPU chỉ để hỏi trạng thái

Thiết bị rất nhanh

Polling



xong

Ngắt



ngắt + đổi ngữ cảnh



xong muộn hơn

vd: SSD NVMe độ trễ thấp

→ chờ ngắn hơn chi phí xử lý ngắt: polling xong trước

LÃNG PHÍ KHI

- Thiết bị chậm, thời gian chờ dài
- CPU còn việc khác để làm
- Phải phục vụ nhiều thiết bị

VẪN HỢP LÝ KHI

- Thời gian chờ ngắn hơn chi phí xử lý ngắt
- Hệ nhúng đơn giản, CPU không có việc khác
- Tải rất cao: Linux NAPI (card mạng), polled I/O (SSD NVMe)

04

PHẦN 04 · MỤC 8.4

I/O điều khiển bằng ngắt

Interrupt-driven I/O · thiết bị lên tiếng khi đã sẵn sàng

Interrupt-driven I/O (I/O ngắt): các bước



AI KHỞI TẠO?

CPU, rồi thiết bị

CPU gửi lệnh; module tự phát ngắt khi sẵn sàng

AI CHỜ?

Không ai đứng chờ

CPU chạy việc khác trong lúc thiết bị làm

AI CHUYỂN DỮ LIỆU?

Vẫn là CPU

Trong ISR (trình phục vụ ngắt), từng word

1 CPU gửi lệnh đọc cho module I/O

2 Module lấy dữ liệu từ thiết bị; CPU chạy chương trình khác

3 Dữ liệu sẵn sàng → module gửi **interrupt** (ngắt) tới CPU

4 CPU chạy **ISR** (trình phục vụ ngắt): chuyển dữ liệu, rồi quay lại việc cũ

Interrupt processing (xử lý một ngắt)

PHẦN CỨNG

- 1 Thiết bị phát tín hiệu ngắt
- 2 CPU làm xong lệnh hiện tại
- 3 CPU báo đã nhận ngắt
- 4 Cất PSW và PC vào ngăn xếp
- 5 Nạp PC = địa chỉ ISR

PHẦN MỀM · ISR

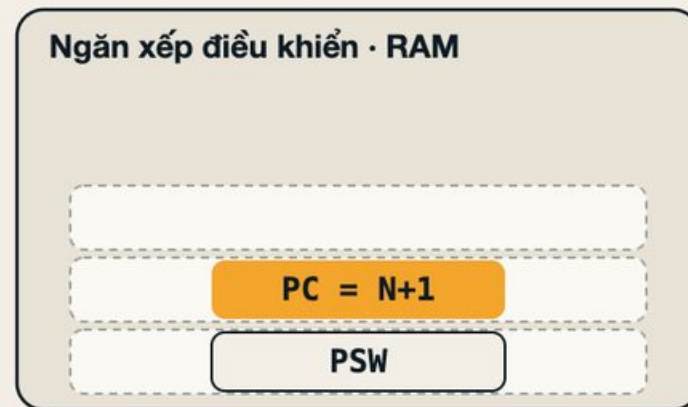
- 6 Lưu các thanh ghi còn lại
- 7 Xử lý ngắt: chuyển dữ liệu
- 8 Khôi phục các thanh ghi
- 9 Khôi phục PSW, PC → chạy tiếp

THUẬT NGỮ

PC (Program Counter): bộ đếm chương trình
PSW (Program Status Word): từ trạng thái
ISR: trình phục vụ ngắt



⑤ PC nhảy tới ISR tại Y



Thiết bị nào vừa gọi ngắt? Bốn cách nhận diện

1 · MULTIPLE INTERRUPT LINES

Nhiều đường ngắt

Mỗi module (hoặc nhóm module) có một đường ngắt riêng tới CPU

Ưu: đơn giản nhất · **Nhược:** CPU có ít đường, mỗi đường vẫn phải dùng chung

2 · SOFTWARE POLL

Hỏi vòng bằng phần mềm

Có ngắt thì ISR chung hỏi lần lượt từng module xem ai đã gọi

Ưu: không cần thêm phần cứng · **Nhược:** tốn thời gian

3 · DAISY CHAIN · CÓ VECTOR

Chuỗi xích (hỏi vòng bằng phần cứng)

Tín hiệu xác nhận chạy qua từng module; module gọi ngắt giữ lại và đặt vector lên bus

Ưu: nhảy thẳng tới đúng ISR · **Nhược:** ưu tiên cố định theo vị trí

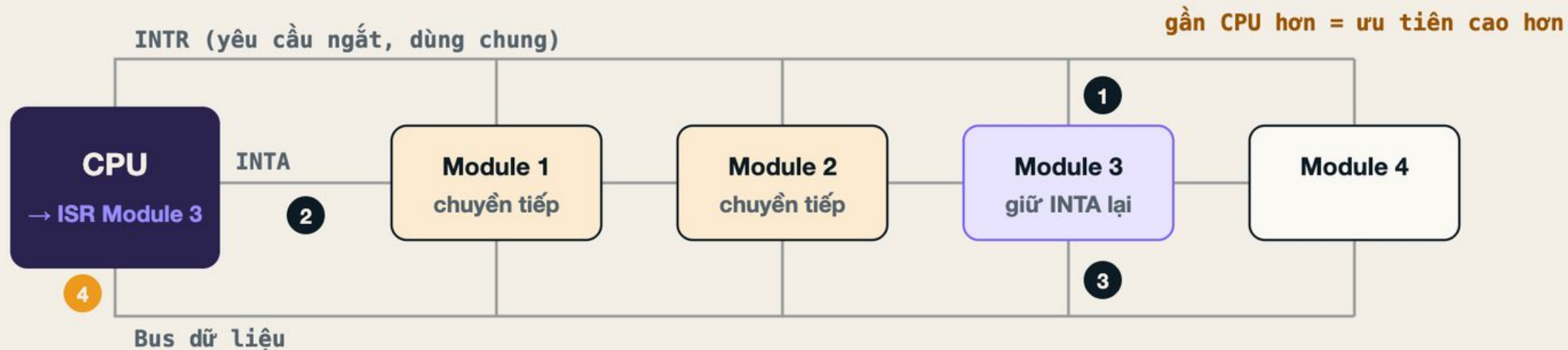
4 · BUS ARBITRATION · CÓ VECTOR

Phân xử bus

Module phải giành quyền dùng bus trước, rồi mới gửi yêu cầu ngắt và đặt vector

Ưu: mỗi lúc chỉ một module gọi · **Nhược:** cần cơ chế phân xử

Daisy chain (chuỗi xích) hoạt động thế nào?



- 1 Module 3 cần phục vụ → kéo đường **INTR** (dùng chung cho mọi module)
- 2 CPU gửi **INTA** (xác nhận ngắt); Module 1, 2 không gọi nên chuyển tiếp
- 3 Module 3 giữ INTA lại, đặt **vector** (mã định danh) lên bus dữ liệu
- 4 CPU dùng vector làm con trỏ, nhảy thẳng tới **ISR** của Module 3

Interrupt priority (ưu tiên ngắt)

↑ ưu tiên cao hơn

Truyền thông · 5

Đĩa · 4

Máy in · 2

Chương trình



t=10 Máy in (ưu tiên 2) ngắt → chạy ISR máy in

t=15 Truyền thông (5) ngắt → cao hơn nên **chen ngang**

t=20 Đĩa (4) ngắt → thấp hơn 5 nên **phải chờ**

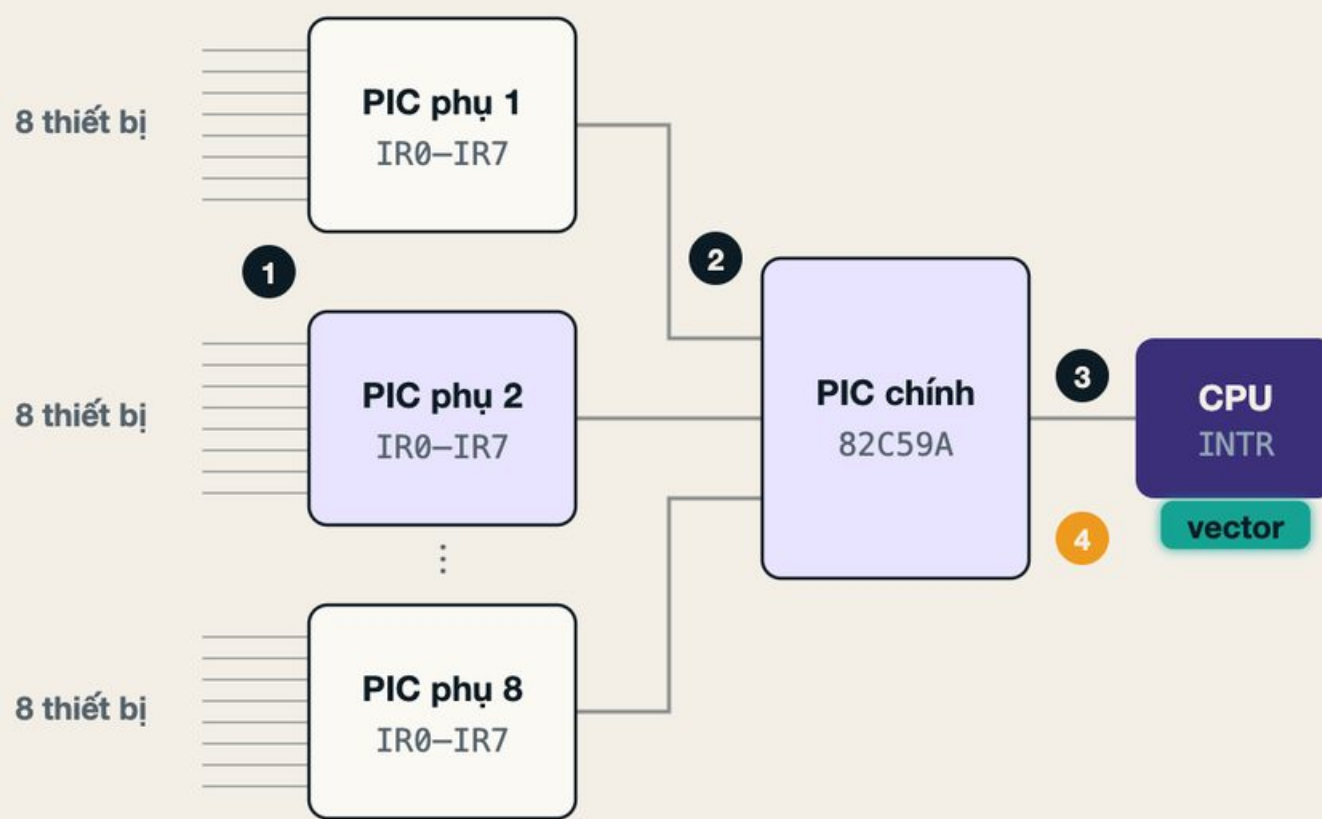
t=25 Xong truyền thông → phục vụ đĩa

t=35 Xong đĩa → quay lại máy in

t=40 Xong máy in → về chương trình

Quy tắc: ngắt ưu tiên cao được chen ngang ISR của ngắt thấp hơn; ngắt thấp hơn phải chờ.

Interrupt controller (bộ điều khiển ngắt)



- 1 Thiết bị kéo một đường **IR** (IR0-IR7) của chip phụ
- 2 Chip phụ báo lên chip chính
- 3 Chip chính chọn yêu cầu **ưu tiên cao nhất**, báo CPU qua đường INTR
- 4 CPU xác nhận; chip gửi **vector** của thiết bị lên bus dữ liệu

1 chip nhận 8 yêu cầu (IR0-IR7)
1 chip chính + 8 chip phụ → tối đa **64 thiết bị**
Mặc định IR0 có ưu tiên cao nhất

Driver (trình điều khiển thiết bị)

PHẦN MỀM

Ứng dụng

gọi read(file)

Hệ điều hành

lời gọi hệ thống, hệ thống tập tin

Driver thiết bị

biết thanh ghi, lệnh, cách xử lý ngắt

PHẦN CỨNG

Bộ điều khiển (module I/O)

thanh ghi lệnh · trạng thái · dữ liệu

Thiết bị

SSD, bàn phím, card mạng...

đọc dữ liệu



Dịch yêu cầu chung thành lệnh riêng

Cùng một read(), mỗi thiết bị cần chuỗi lệnh khác nhau



Làm việc với thanh ghi

Ghi thanh ghi lệnh, đọc thanh ghi trạng thái của bộ điều khiển



Chứa trình phục vụ ngắt

Khi thiết bị báo xong, ISR của driver xử lý kết quả

05

PHẦN 05 · MỤC 8.5

Truy cập bộ nhớ trực tiếp

Direct Memory Access (DMA) · giao cả khối dữ liệu cho bộ điều khiển DMA

Vì sao cần DMA?

- 1 Tốc độ truyền bị giới hạn**
bởi tốc độ CPU kiểm tra và phục vụ thiết bị

- 2 CPU bị trói vào việc truyền**
mỗi lần truyền phải chạy nhiều lệnh

I/O ngắt

mỗi word đi qua CPU



số lần ngắt

315

DMA

cả khối vào thẳng RAM

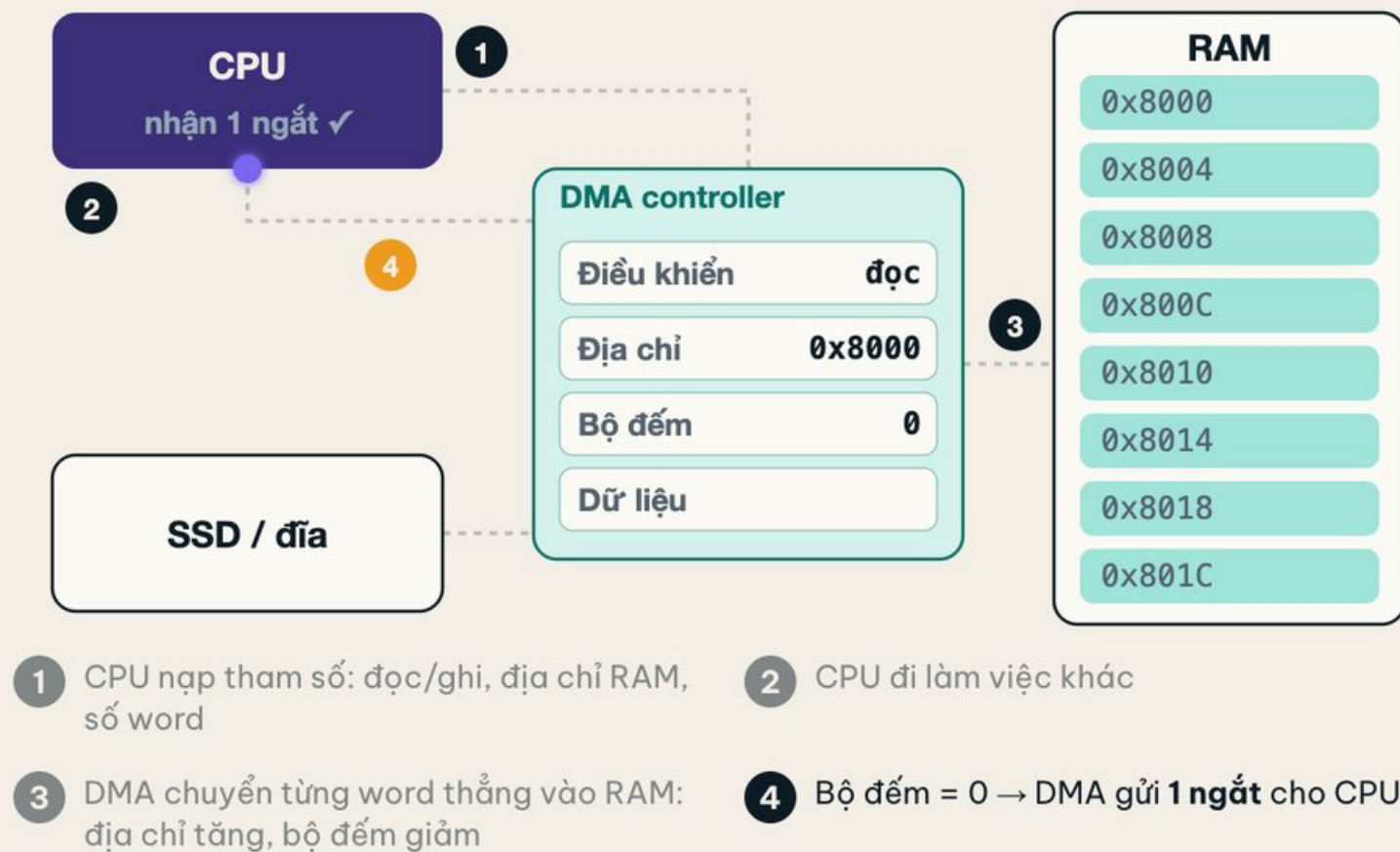


số lần ngắt

0

Ví dụ: khối 4 KB, word 4 byte → 1 024 word

DMA: các bước chuyển một khối dữ liệu



AI KHỞI TẠO?

CPU

Lập trình DMA controller: hướng, địa chỉ, số word

AI CHỜ?

Không ai đứng chờ

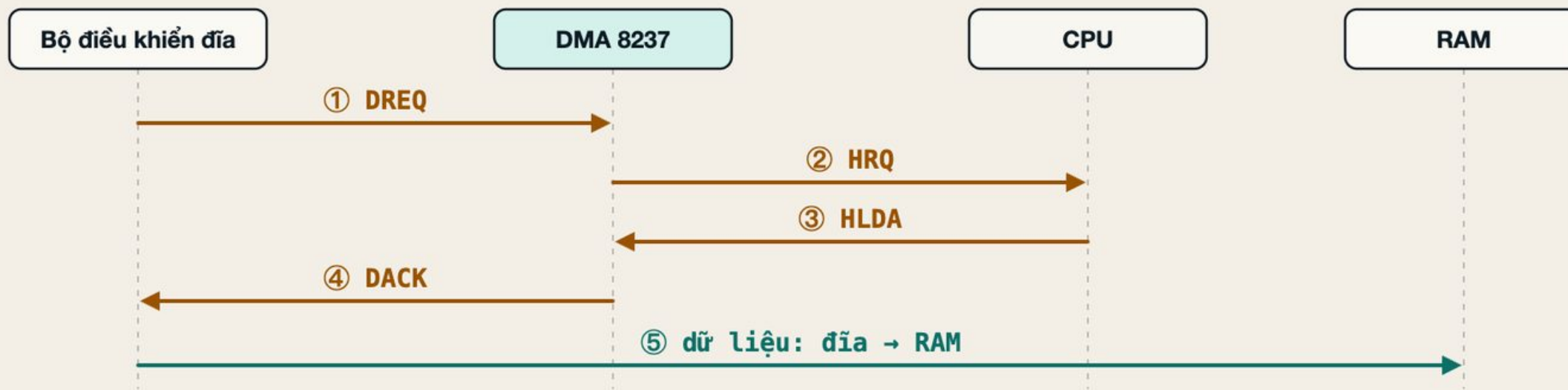
CPU làm việc khác, chỉ nhường bus khi DMA cần

AI CHUYỂN DỮ LIỆU?

DMA controller

(bộ điều khiển DMA) chuyển thẳng thiết bị ↔ RAM

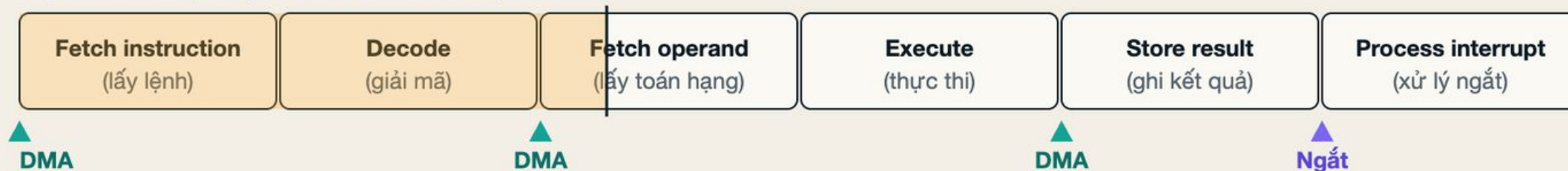
DMA mượn bus: năm bước bắt tay



- ① **DREQ** (DMA request): bộ điều khiển đĩa xin chuyển dữ liệu
- ② **HRQ** (hold request): chip DMA xin CPU nhường bus
- ③ **HLDA** (hold acknowledge): CPU đồng ý, tạm nhường bus
- ④ **DACK** (DMA acknowledge): DMA báo đĩa bắt đầu
- ⑤ Dữ liệu đi thẳng đĩa → RAM, không qua chip DMA (fly-by)

Cycle stealing (lấy cắp chu kỳ)

MỘT CHU KỲ LỆNH (INSTRUCTION CYCLE)



Mượn 1 chu kỳ bus

DMA dùng bus một chu kỳ rồi trả lại. CPU chỉ tạm dừng, **không phải** lưu trữ cảnh.

Chen vào giữa lệnh

DMA breakpoint (tam giác xanh): nhiều điểm ngay trong một chu kỳ lệnh.

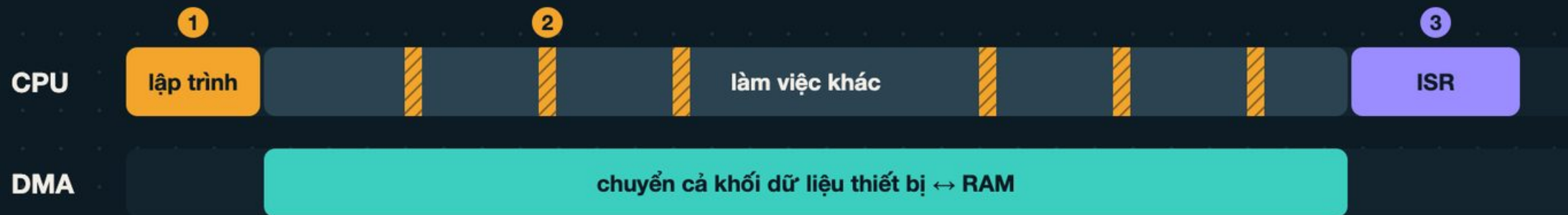
Ngắt thì khác

Interrupt breakpoint (tam giác tím): chỉ kiểm tra ở **cuối** chu kỳ lệnh.

Q2

DMA có chuyển dữ liệu mà không cần CPU chút nào?

Không hẳn. CPU không chạm vào từng byte, nhưng vẫn có mặt ở ba thời điểm.



① Đầu: lập trình DMA

CPU (driver) nạp địa chỉ, số word, hướng truyền và chuẩn bị bộ đệm

② Giữa: nhường chu kỳ bus

Cycle stealing (lấy cắp chu kỳ): CPU có thể phải tạm dừng khi cũng cần bus

③ Cuối: ngắt báo xong

CPU chạy ISR kiểm tra kết quả và đánh thức tiến trình đang chờ

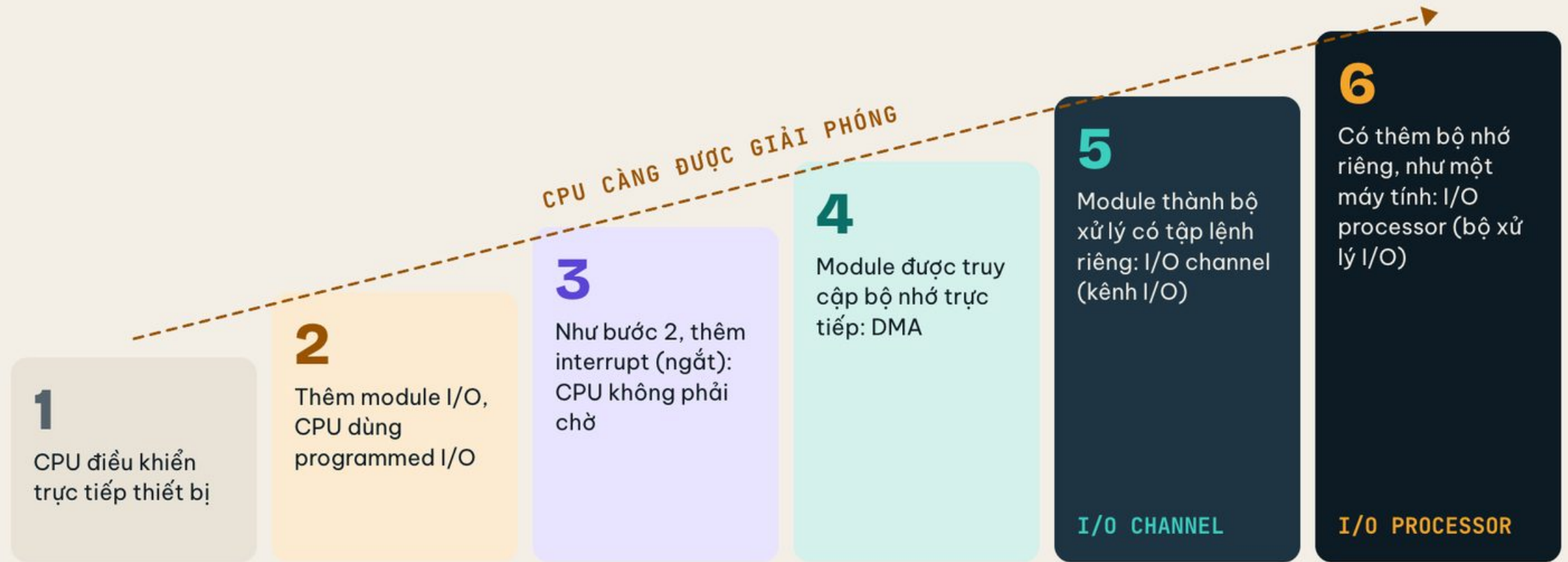
06

PHẦN 06 · MỤC 8.6

Kênh và bộ xử lý I/O

I/O Channels & Processors · giao hửn việc I/O cho một bộ xử lý riêng

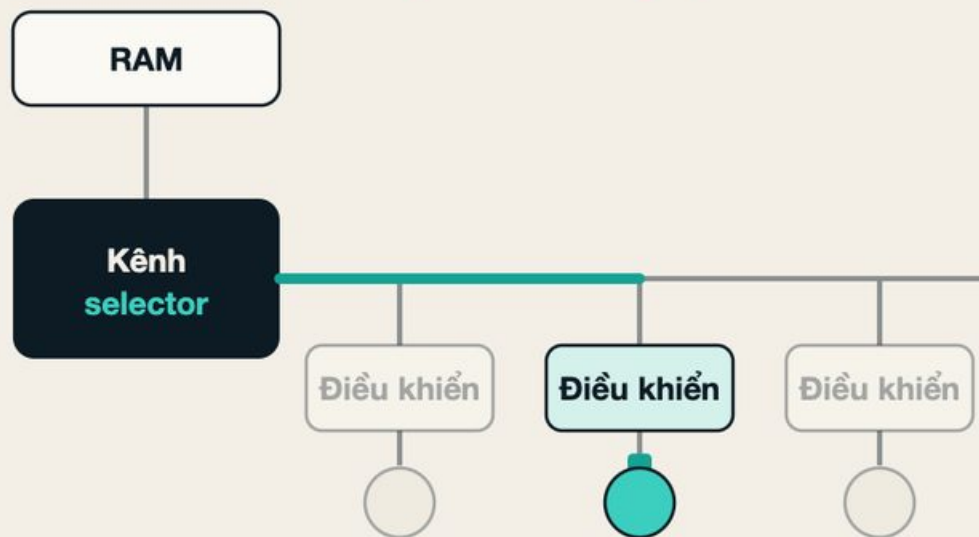
Sáu bước tiến hoá của chức năng I/O



Kênh I/O: selector và multiplexor

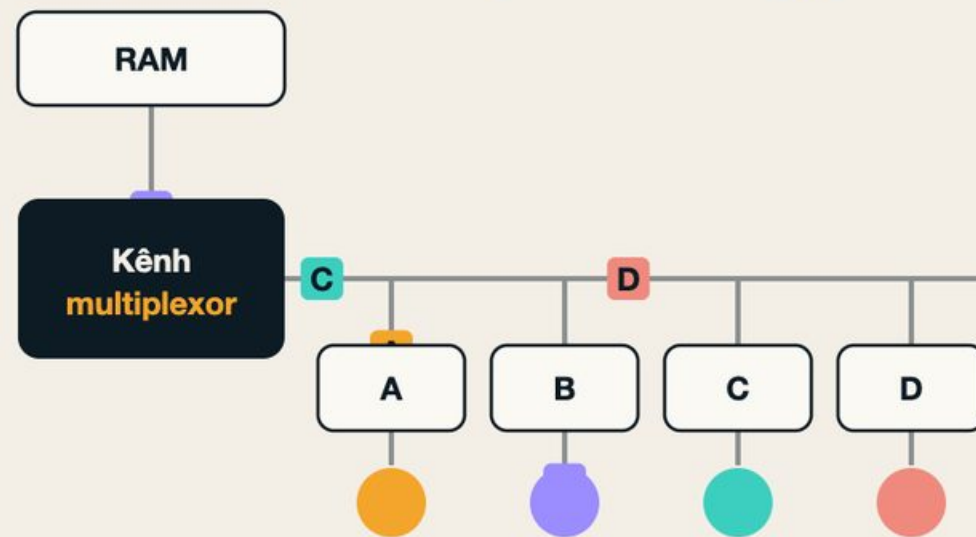
- 1 CPU viết **channel program** (chương trình kênh) vào RAM
- 2 CPU ra lệnh cho kênh bắt đầu chạy
- 3 Kênh tự thực hiện, xong thì **ngắt** CPU

Selector channel (kênh chọn)



Mỗi lúc phục vụ **một** thiết bị tốc độ cao, trọn một khối

Multiplexor channel (kênh đa hợp)



Xen kẽ dữ liệu của **nhiều** thiết bị tốc độ thấp

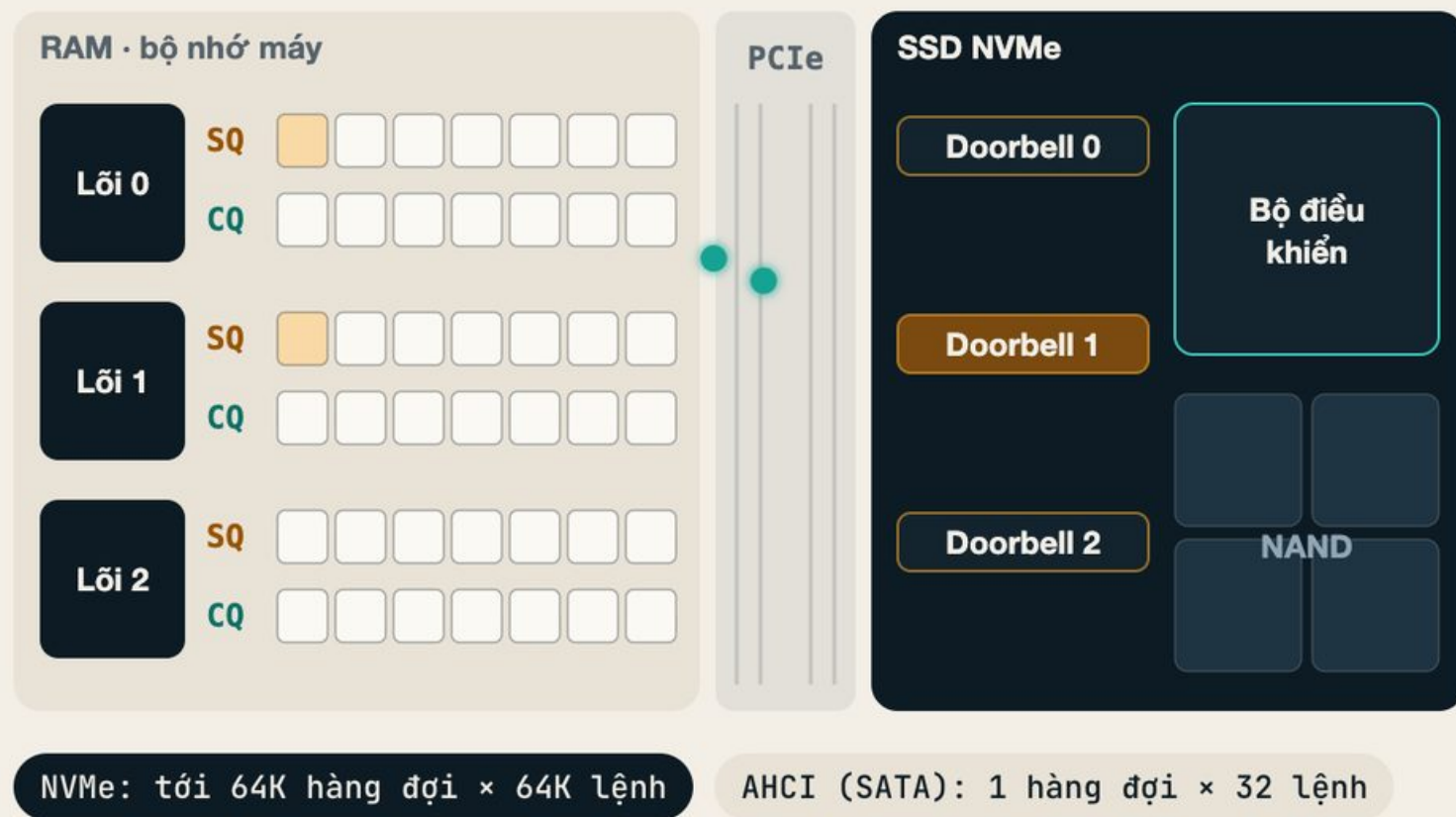
07

PHẦN 07 · LIÊN HỆ HIỆN ĐẠI

NVMe SSD trên PCIe

Hàng đợi lệnh · DMA · thông báo hoàn tất trong một chiếc SSD hiện đại

NVMe: mỗi lõi CPU một cặp hàng đợi



SQ • SUBMISSION QUEUE

Hàng đợi gửi lệnh

Vòng đếm trong RAM, mỗi lệnh 64 byte

DOORBELL

Thanh ghi “chuông cửa”

Nằm trên SSD; CPU ghi vào để báo có lệnh mới

CQ • COMPLETION QUEUE

Hàng đợi hoàn tất + MSI-X

SSD ghi kết quả rồi gửi ngắt; hoặc CPU tự hỏi vòng CQ

Q3

Từ lệnh `read()` đến khi dữ liệu SSD nằm trong RAM: chuyện gì xảy ra?



Tám bước, ba khái niệm của NVMe

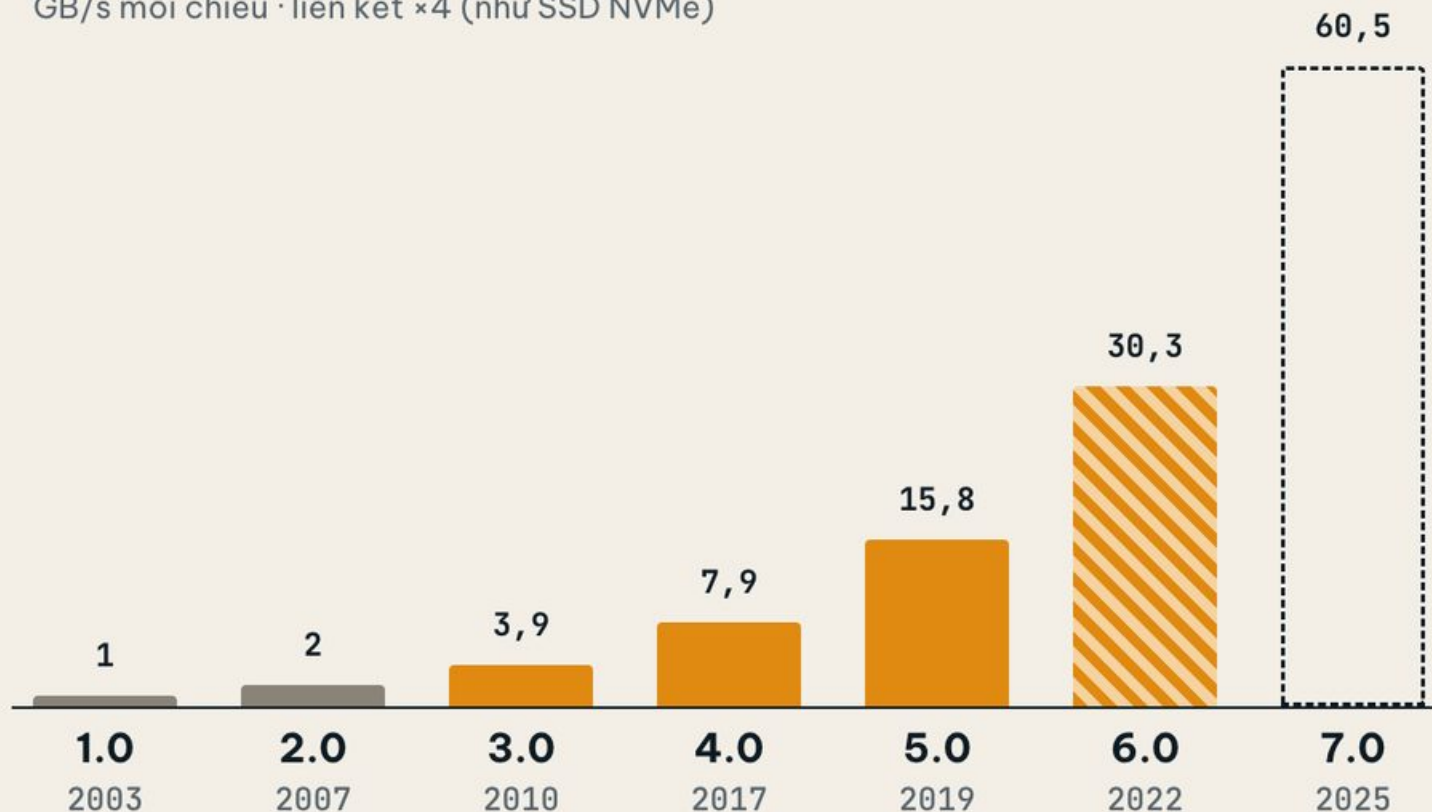
● **Command queue** (hàng đợi lệnh) ● **DMA** (truy cập bộ nhớ trực tiếp) ● **Completion** (thông báo hoàn tất)

- 1 Ứng dụng gọi **read()** → CPU chuyển vào nhân hệ điều hành
- 2 Driver cấp bộ đệm trong RAM, ghi lệnh Read 64 byte vào **SQ** (submission queue)
- 3 CPU ghi **doorbell** (thanh ghi chuông cửa) trên SSD: “có lệnh mới”
- 4 SSD dùng **DMA** đọc lệnh từ SQ trong RAM
- 5 SSD đọc dữ liệu từ chip **NAND** (tra bảng ánh xạ FTL)
- 6 SSD dùng **DMA** ghi dữ liệu thẳng vào bộ đệm trong RAM
- 7 SSD ghi kết quả vào **CQ** (completion queue), gửi ngắt MSI-X
- 8 Driver đọc CQ, đánh thức tiến trình → **read()** trả về

CPU chỉ làm bước 1–3 và 8. Phần đọc và chuyển dữ liệu (bước 4–7) do **SSD tự làm bằng DMA**.

PCIe: băng thông gấp đôi mỗi thế hệ

GB/s mỗi chiều · liên kết ×4 (như SSD NVMe)



- Thế hệ cũ
- Phổ biến trong PC hiện nay
- Bắt đầu có ở máy chủ
- Đã công bố, chưa phổ biến

Công bố chuẩn ≠ máy tính đã dùng

CPU, bo mạch chủ và thiết bị phải cùng hỗ trợ; liên kết chạy ở thế hệ cao nhất mà cả hai đầu đều có.

Q4

Bus nhanh hơn có luôn giảm thời gian đáp ứng cho I/O nhỏ?

$T \approx \text{độ trễ cố định} + \text{kích thước} \div \text{băng thông}$

Số liệu làm tròn

Đọc 4 KB · I/O nhỏ



Đọc 1 GB · I/O lớn (giả sử SSD đủ nhanh)



Không. Với I/O nhỏ, độ trễ cố định chiếm gần hết thời gian; bus nhanh hơn chỉ thắng rõ khi dữ liệu lớn.

08

PHẦN 08 • TỔNG KẾT

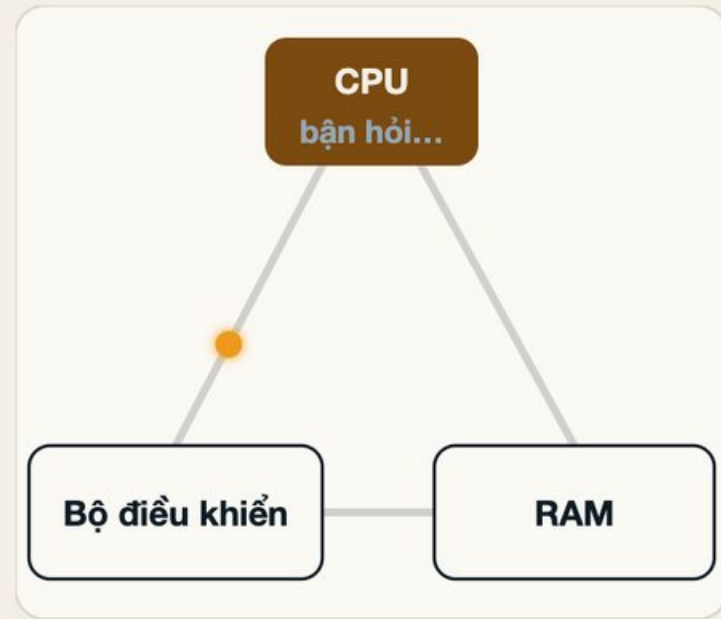
So sánh và tổng kết

Ba con đường giữa CPU, bộ điều khiển và RAM



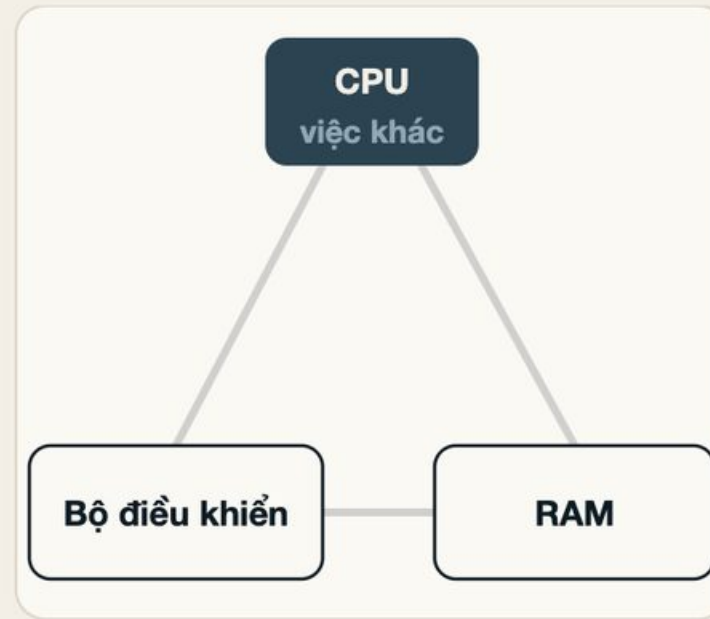
Ba con đường: CPU, bộ điều khiển, RAM

Polling (hỏi vòng)



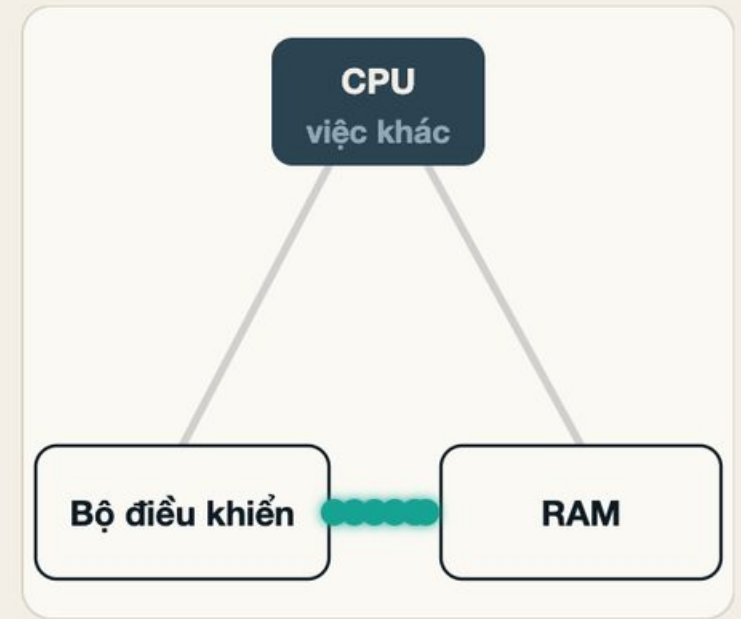
Dữ liệu đi: **bộ điều khiển** → CPU → RAM
CPU: **luôn bận hỏi, 0 ngắt**

Interrupt (ngắt)



Dữ liệu đi: **bộ điều khiển** → CPU → RAM
CPU: **rảnh, nhưng 1 ngắt mỗi word**

DMA



Dữ liệu đi: **bộ điều khiển** → RAM
CPU: **rảnh, 1 ngắt mỗi khối**

Ai khởi tạo, ai chờ, ai chuyển dữ liệu?

Tiêu chí	Programmed I/O (I/O lập trình)	Interrupt-driven I/O (I/O ngắt)	DMA (truy cập trực tiếp)
Ai khởi tạo?	CPU	CPU gửi lệnh; thiết bị báo bằng ngắt	CPU lập trình bộ DMA
Ai chờ?	CPU: busy waiting (chờ bận)	Không ai; CPU làm việc khác	Không ai; CPU chỉ nhường bus
Ai chuyển dữ liệu?	CPU, từng word	CPU trong ISR (trình phục vụ ngắt), từng word	DMA controller (bộ điều khiển DMA)
Số lần ngắt	0	Mỗi word một lần	Một lần mỗi khối
Phù hợp khi	Thiết bị rất nhanh, hệ đơn giản	Thiết bị chậm, dữ liệu ít	Khối dữ liệu lớn: đĩa, mạng

Cảm ơn!

Hỏi & đáp

Tài liệu: W. Stallings, *Computer Organization and Architecture*, 11e,
Chương 8 · NVM Express Base Specification · PCI-SIG

Nhóm 4 · Fall 2026

