KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG VÀ CÁC HỆ THỐNG THÔNG TIN

CHƯƠNG 3

BỘ XỬ LÝ TRUNG TÂM

Mục tiêu

Sau khi hoàn thành chương này, sinh viên có khả năng:

- Trình bày được nhiệm vụ và cấu trúc cơ bản của CPU.
- Mô tả được nguyên lý hoạt động của CPU.
- Phân tích được kiến trúc của các bộ xử lý tiên tiến.
- Giải thích được tập lệnh cơ bản của bộ xử lý 8086.

Nội dung

3.1. Nhiệm vụ và cấu trúc cơ bản của CPU

3.2. Hoạt động của CPU

3.3. Kiến trúc các bộ xử lý tiên tiến

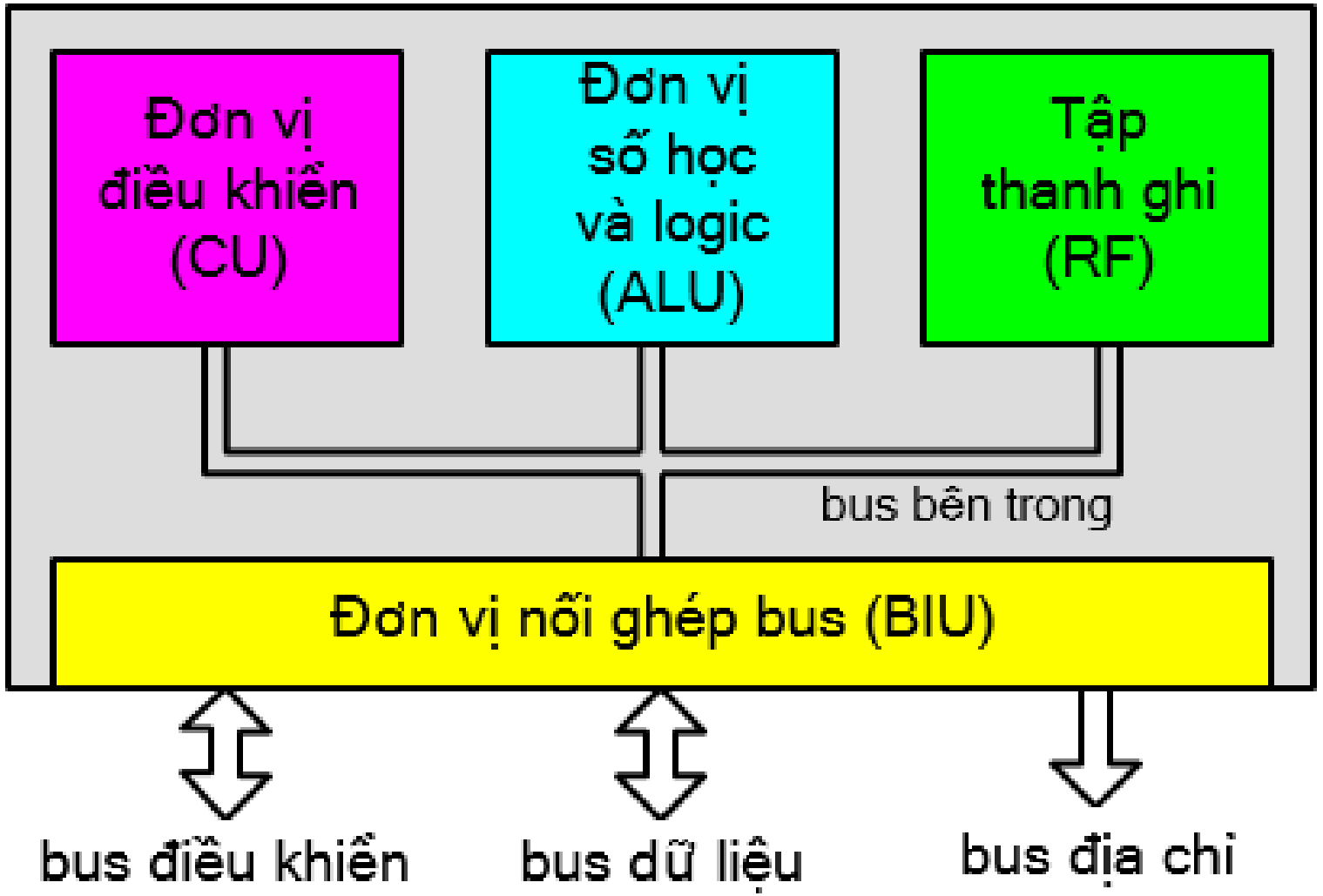
3.4. Tập lệnh của 8086

3.1. Nhiệm vụ và cấu trúc cơ bản của CPU

3.1.1. 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): nhận dữ liệu từ bộ nhớ hoặc các cổng vào-ra
- Xử lý dữ liệu (Process Data): thực hiện phép toán số học hay phép toán logic với các dữ liệu
- Ghi dữ liệu (Write Data): ghi dữ liệu ra bộ nhớ hay cổng vào-ra

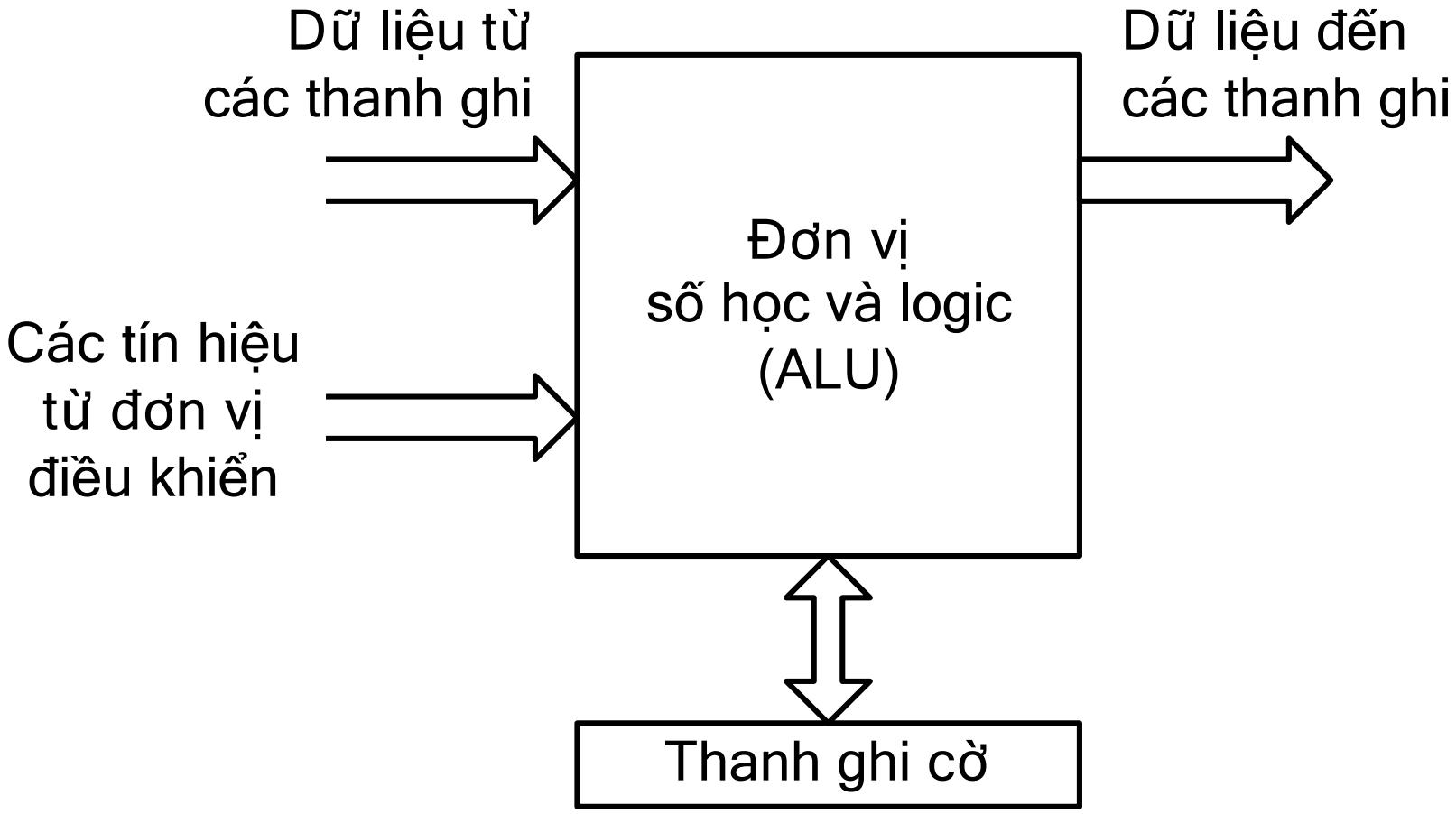
Sơ đồ cấu trúc cơ bản của CPU



Đơn vị số học và logic

- Chức năng: 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

Mô hình kết nối ALU

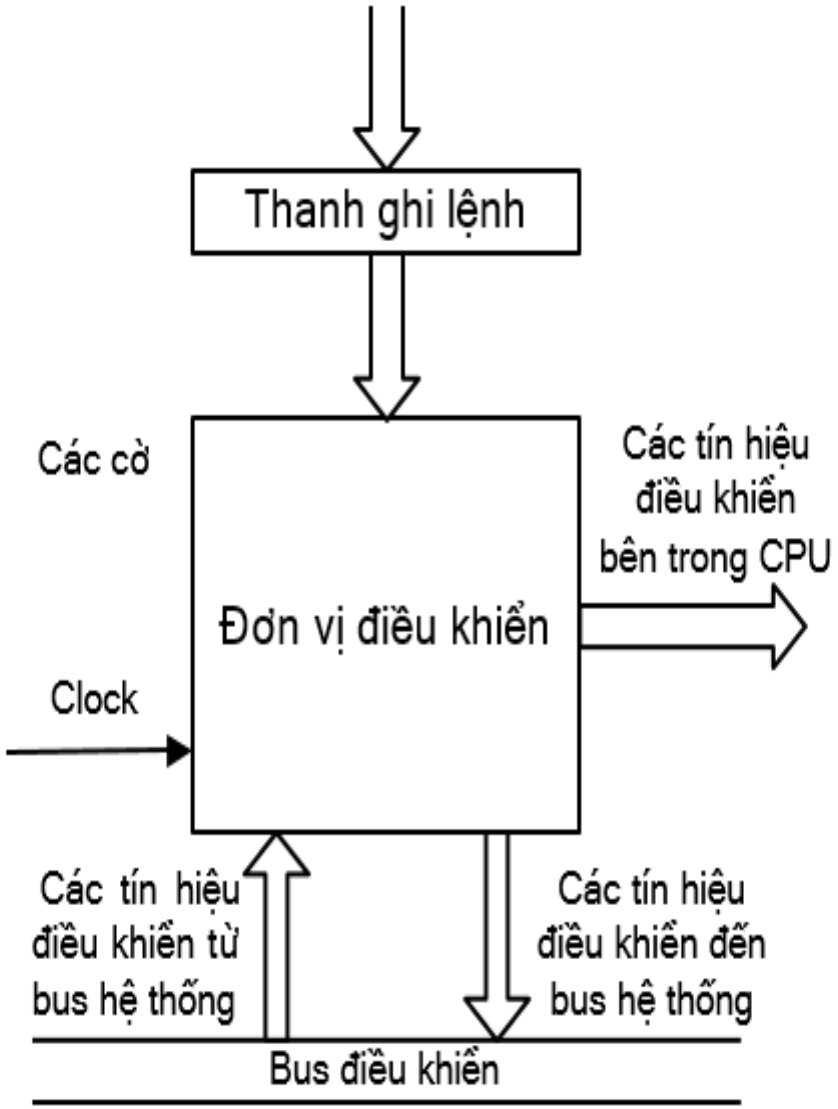


Thanh ghi cờ: hiển thị trạng thái của kết quả phép toán

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 hệ thống và đáp ứng với các yêu cầu đó.

Mô hình kết nối đơn vị điều khiển



Các tín hiệu đưa đến đơn vị điều khiển

- 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

Các tín hiệu phát ra từ đơn vị điều khiển

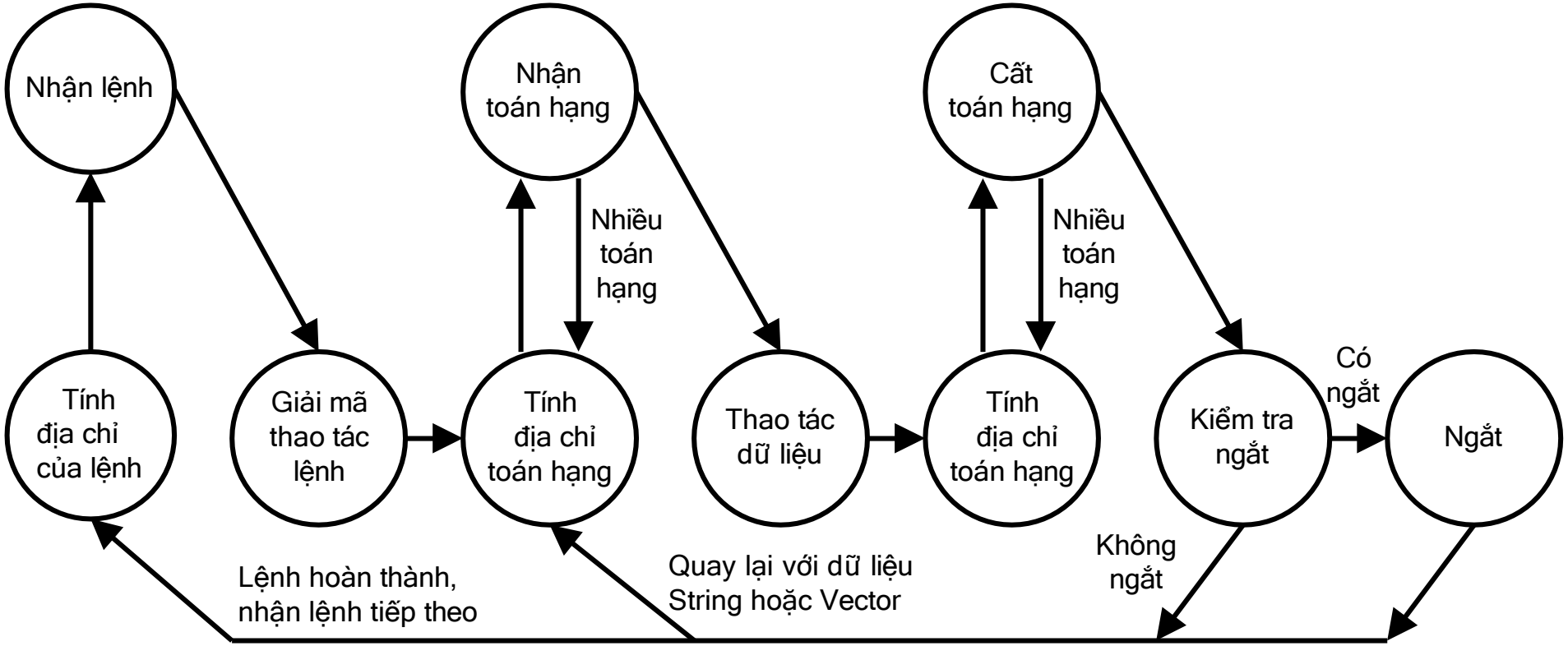
- Các tín hiệu điều khiển bên trong CPU:
  - Điều khiển các thanh ghi
  - Điều khiển ALU
- Các tín hiệu điều khiển bên ngoài CPU:
  - Điều khiển bộ nhớ
  - Điều khiển các mô-đun vào-ra

Hoạt động của chu trình lệnh

Chu trình lệnh

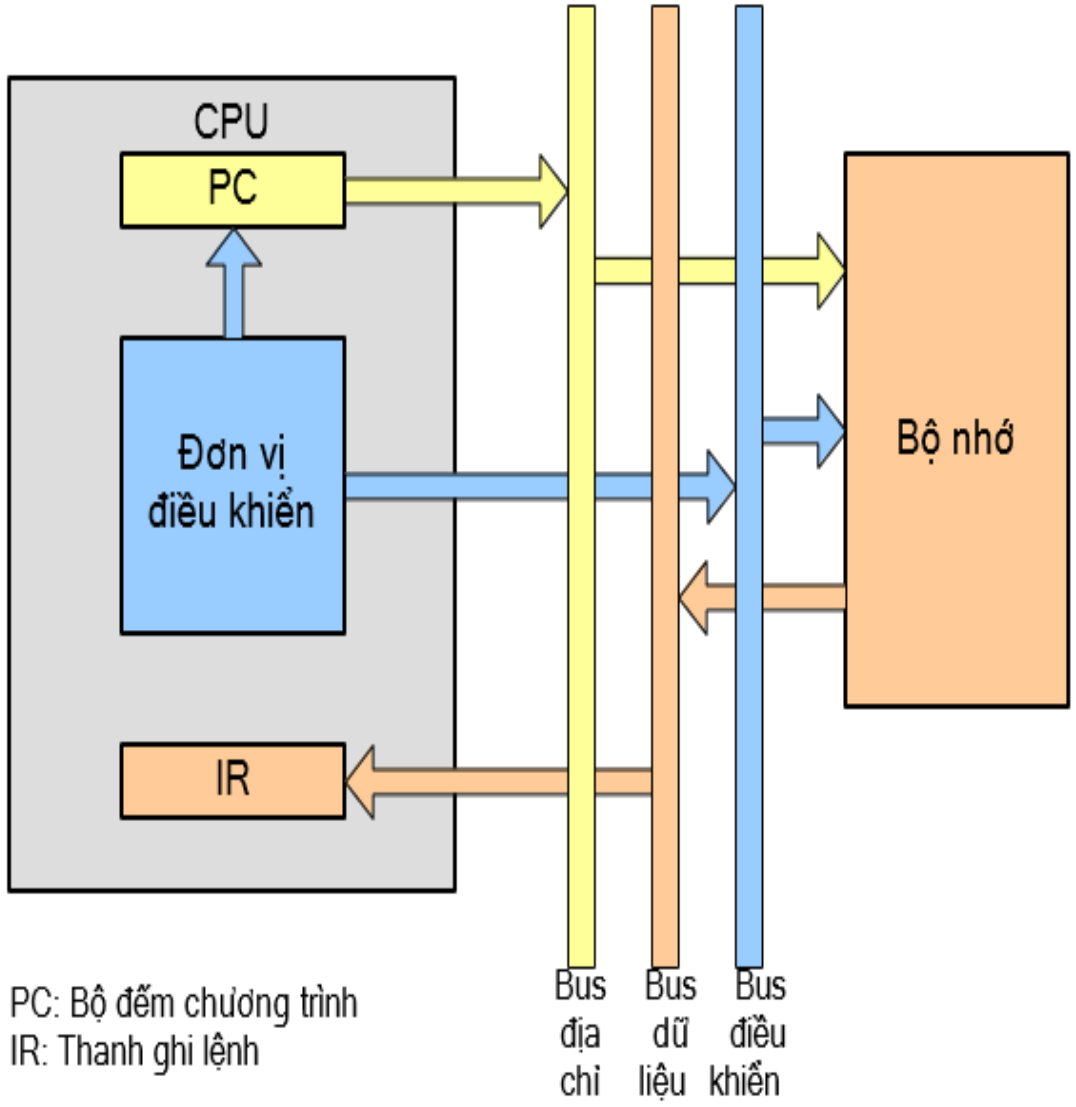
- Nhận lệnh
- Giải mã lệnh
- Nhận toán hạng
- Thực hiện lệnh
- Cất toán hạng
- Ngắt

Giản đồ trạng thái chu trình lệnh



Nhận lệnh

- CPU đưa địa chỉ của lệnh cần nhận từ bộ đếm chương trình PC ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc bộ nhớ
- Lệnh từ bộ nhớ được đặt lên bus dữ liệu và được CPU copy vào thanh ghi lệnh IR
- CPU tăng nội dung PC để trỏ sang lệnh kế tiếp

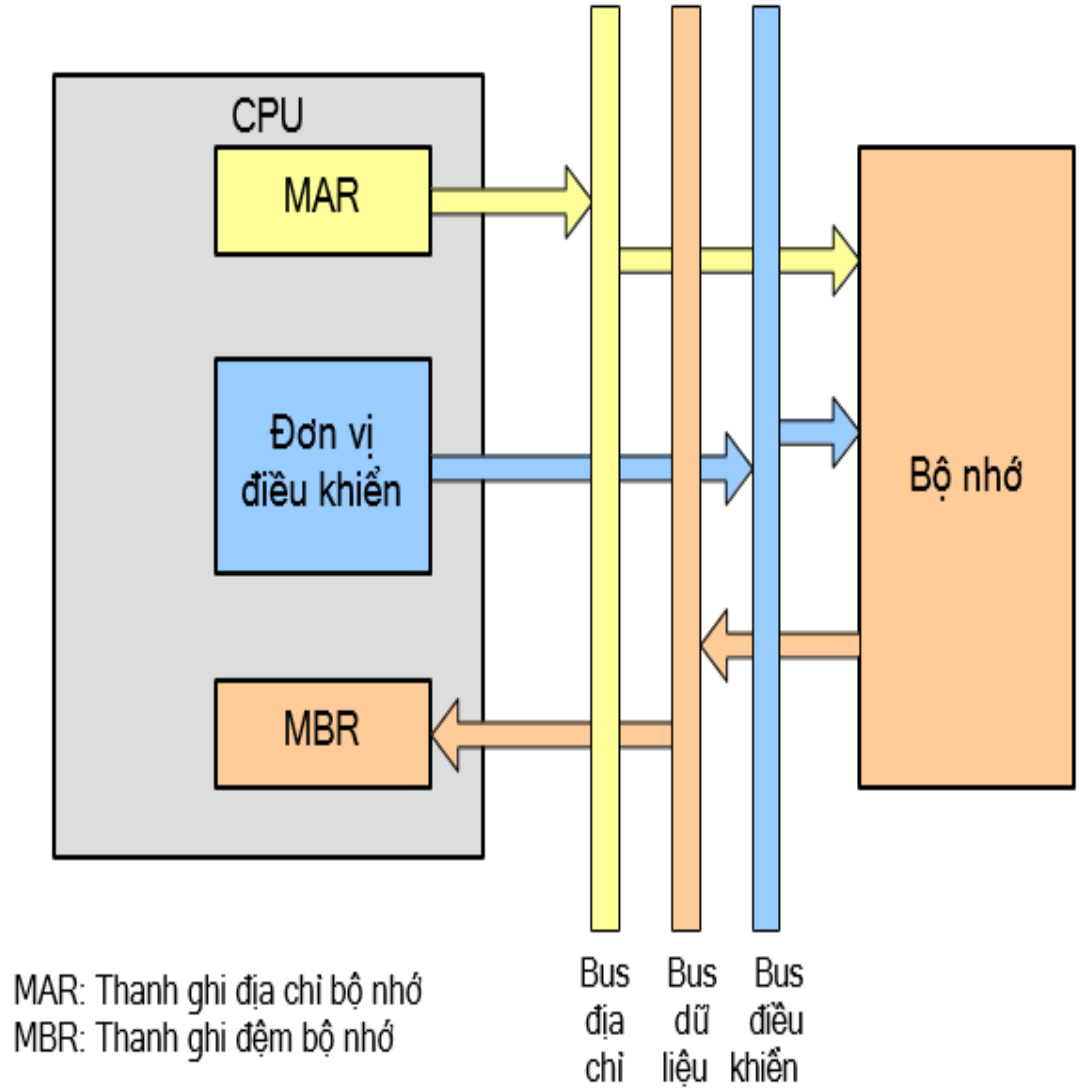


Giải mã lệnh

- Lệnh từ thanh ghi lệnh IR được đưa đến đơn vị điều khiển
- Đơn vị điều khiển tiến hành giải mã lệnh để xác định thao tác phải thực hiện
- Giải mã lệnh xảy ra bên trong CPU

Nhận dữ liệu từ bộ nhớ

- CPU đưa địa chỉ của toán hạng ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc
- Toán hạng được đọc vào CPU
- Tương tự như nhận lệnh



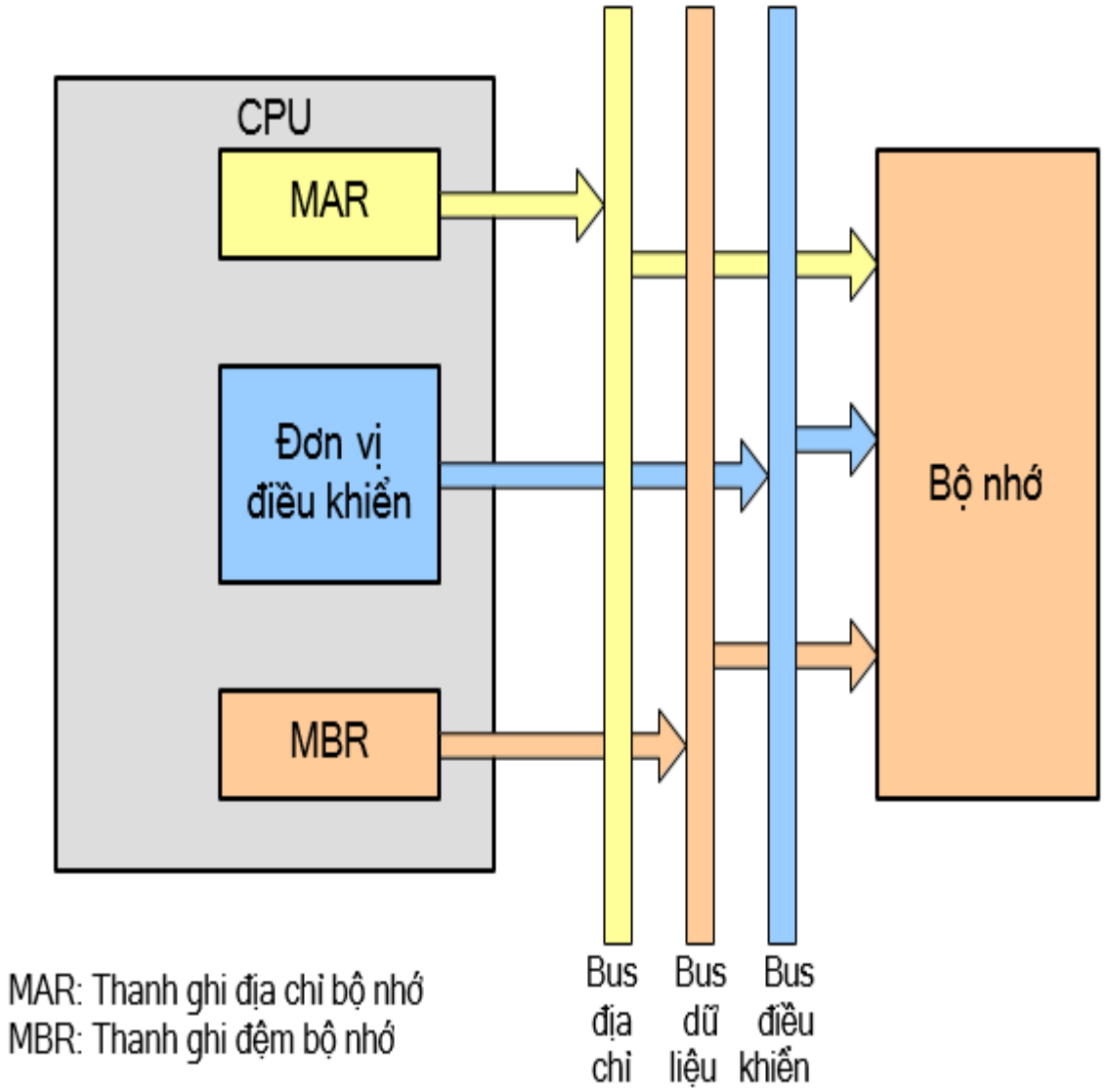
MAR: Thanh ghi địa chỉ bộ nhớ
MBR: Thanh ghi đệm bộ nhớ

Thực hiện lệnh

- Có nhiều dạng tuỳ thuộc vào lệnh
- Có thể là:
  - Đọc/Ghi bộ nhớ
  - Vào/Ra
  - Chuyển giữa các thanh ghi
  - Phép toán số học/logic
  - Chuyển điều khiển (rẽ nhánh)
  - ...

Ghi toán hạng

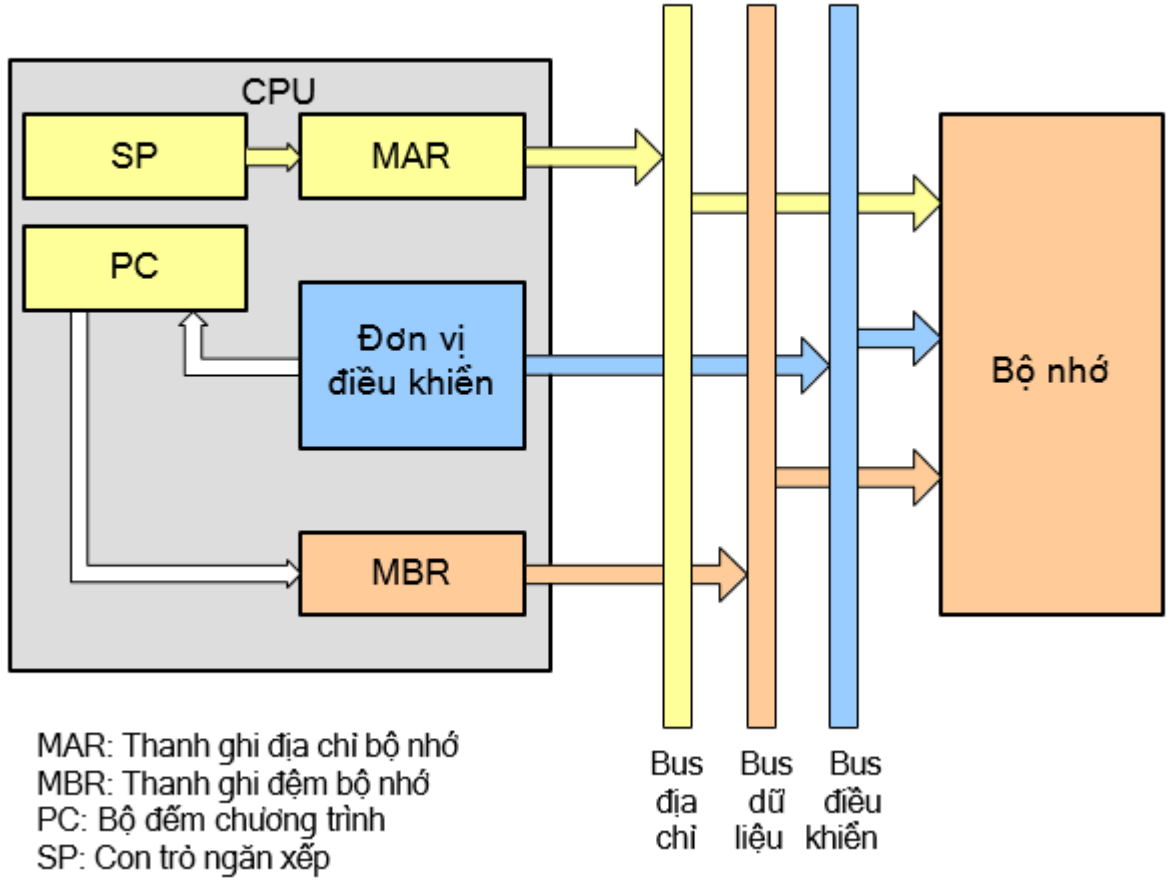
- CPU đưa địa chỉ ra bus địa chỉ
- CPU đưa dữ liệu cần ghi ra bus dữ liệu
- CPU phát tín hiệu điều khiển ghi
- Dữ liệu trên bus dữ liệu được copy đến vị trí xác định



Ngắt

- Nội dung của bộ đếm chương trình PC (địa chỉ trở về sau khi ngắt) được đưa ra bus dữ liệu
- CPU đưa địa chỉ (thường được lấy từ con trỏ ngăn xếp SP) ra bus địa chỉ
- CPU phát tín hiệu điều khiển ghi bộ nhớ
- Địa chỉ trở về trên bus dữ liệu được ghi ra vị trí xác định (ở ngăn xếp)
- Địa chỉ lệnh đầu tiên của chương trình con điều khiển ngắt được nạp vào PC

Sơ đồ mô tả chu trình ngắt



3.3. Kiến trúc các bộ xử lý tiên tiến

3.3.1. Kiến trúc CISC và RISC

CISC (Complex Instruction Set Computer)

- **Đặc điểm: Tập lệnh phức tạp, mỗi lệnh có thể thực hiện nhiều thao tác.**
- **Ưu điểm:**
  - **Mật độ mã cao, chương trình ngắn gọn hơn.**
  - **Dễ dàng lập trình bằng hợp ngữ (vì có các lệnh phức tạp hơn).**
- **Nhược điểm:**
  - **Chu kỳ lệnh không cố định, khó tối ưu pipeline.**
  - **CPU phức tạp, tốn nhiều transistor hơn.**
  - **Tiêu thụ điện năng cao hơn.**

Ví dụ: Intel x86 (mặc dù các CPU x86 hiện đại sử dụng vi kiến trúc RISC bên trong).

3.3. Kiến trúc các bộ xử lý tiên tiến

3.3.1. Kiến trúc RISC và CISC

RISC (Reduced Instruction Set Computer)

- **Đặc điểm: Tập lệnh đơn giản, ít lệnh, mỗi lệnh thực hiện một thao tác duy nhất.**
- **Ưu điểm:**
  - **Chu kỳ lệnh cố định, dễ dàng tối ưu hóa pipeline.**
  - **CPU đơn giản hơn, tốn ít transistor hơn.**
  - **Tiêu thụ điện năng thấp hơn, tốc độ cao hơn trên cùng một tần số xung nhịp.**
- **Nhược điểm:**
  - **Mật độ mã thấp hơn, chương trình dài hơn.**
  - **Đòi hỏi trình biên dịch phải tối ưu hóa tốt hơn.**

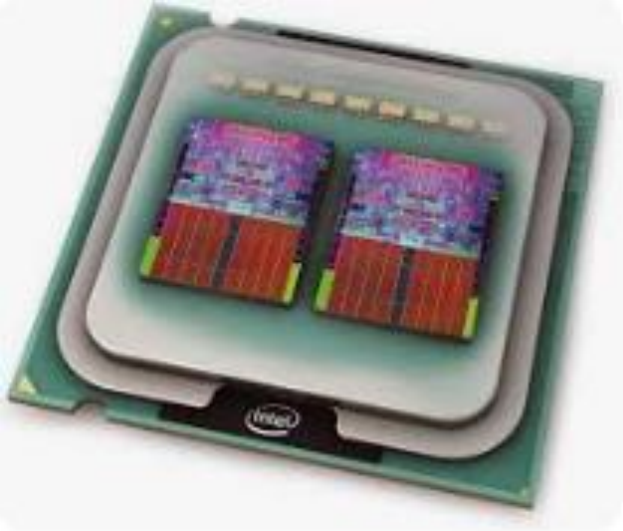
Ví dụ: ARM, MIPS, SPARC

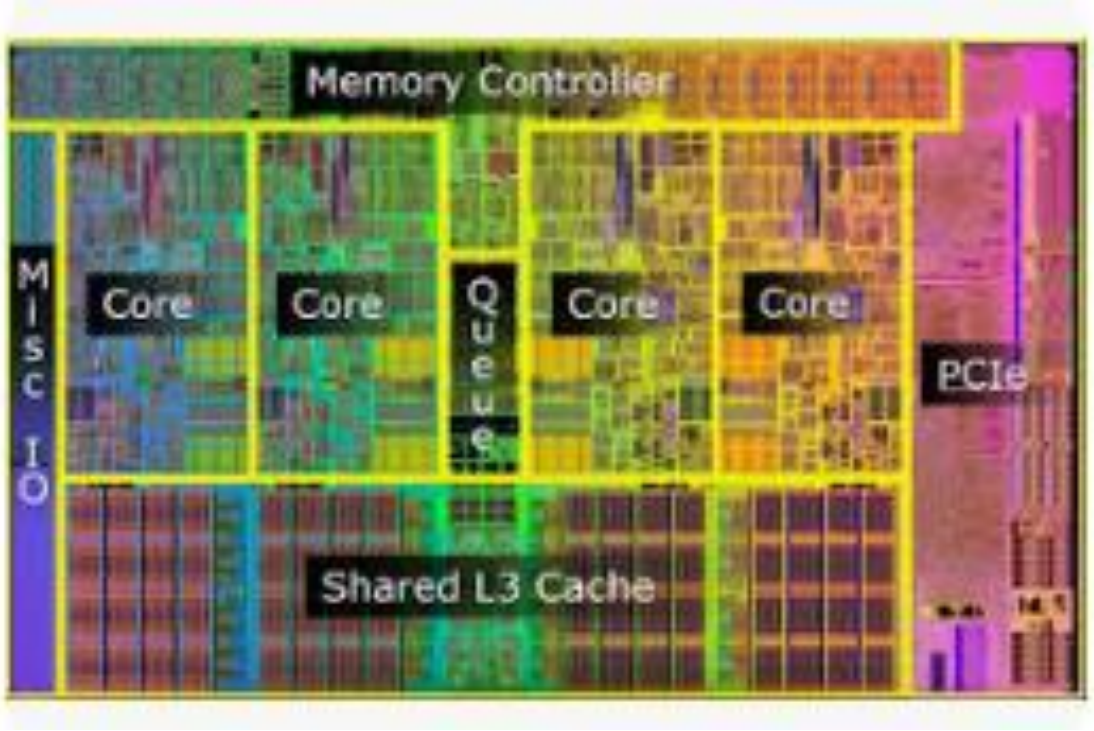
3.3. Kiến trúc các bộ xử lý tiên tiến

3.3.2. Kiến trúc Đa lõi (Multi-core Architecture)

- **Khái niệm: Một chip CPU chứa nhiều "lõi" xử lý hoàn chỉnh, mỗi lõi có CU, ALU và các thanh ghi riêng.**
- **Lợi ích: Tăng khả năng xử lý song song, cho phép chạy nhiều chương trình hoặc nhiều luồng của cùng một chương trình cùng lúc.**

Ví dụ: CPU Core i3, i5, i7, i9 của Intel, Ryzen của AMD.



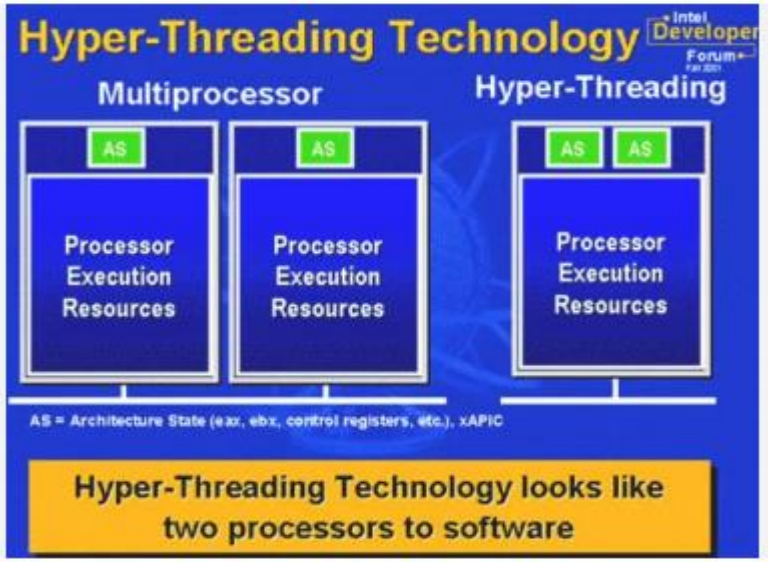


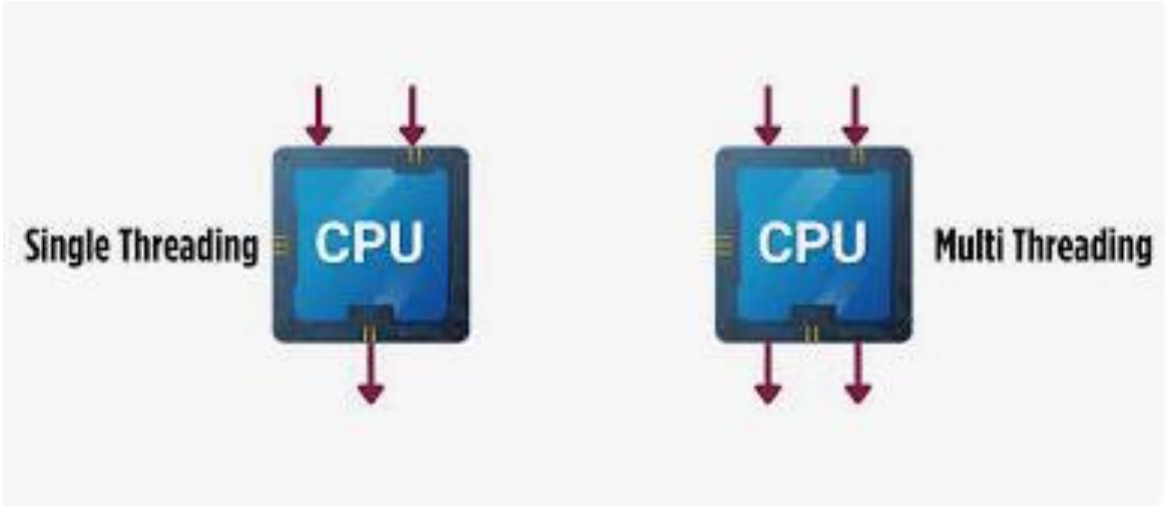
3.3. Kiến trúc các bộ xử lý tiên tiến

3.3.3. Kiến trúc Hyper-Threading (Intel) / SMT (Simultaneous Multithreading)

- **Khái niệm:** Kỹ thuật cho phép một lõi vật lý của CPU giả lập thành nhiều lõi logic (luồng).
- **Cách hoạt động:** Khi một luồng đang chờ dữ liệu (ví dụ: từ bộ nhớ), lõi đó có thể chuyển sang xử lý một luồng khác, tận dụng tối đa các đơn vị thực thi nhàn rỗi.
- **Lợi ích:** Tăng hiệu suất sử dụng tài nguyên của CPU, cải thiện hiệu suất cho các ứng dụng có nhiều luồng.

Lưu ý: Không giống như đa lõi, Hyper-Threading không tăng số lượng đơn vị thực thi vật lý, mà chỉ tối ưu hóa việc sử dụng chúng.

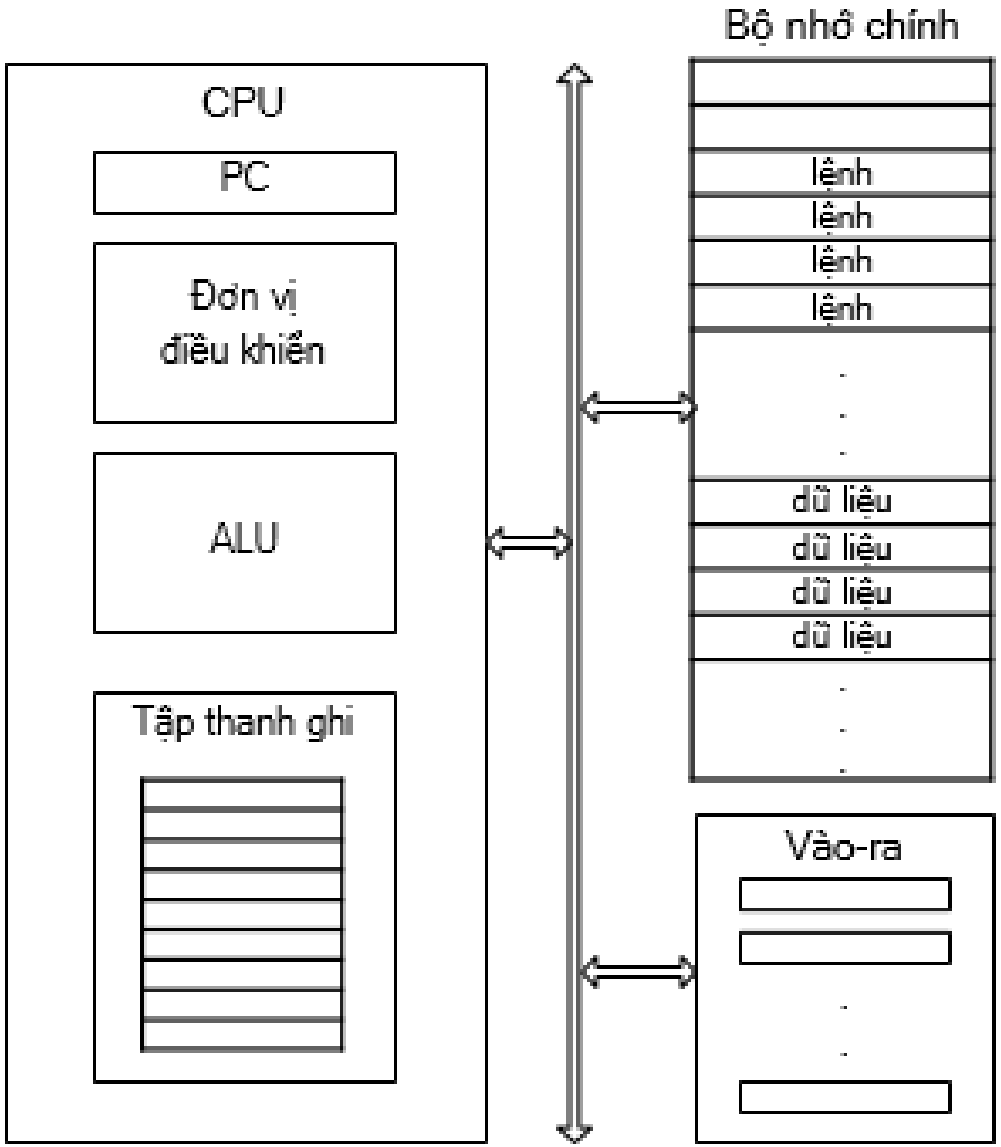




Giới thiệu chung về kiến trúc tập lệnh

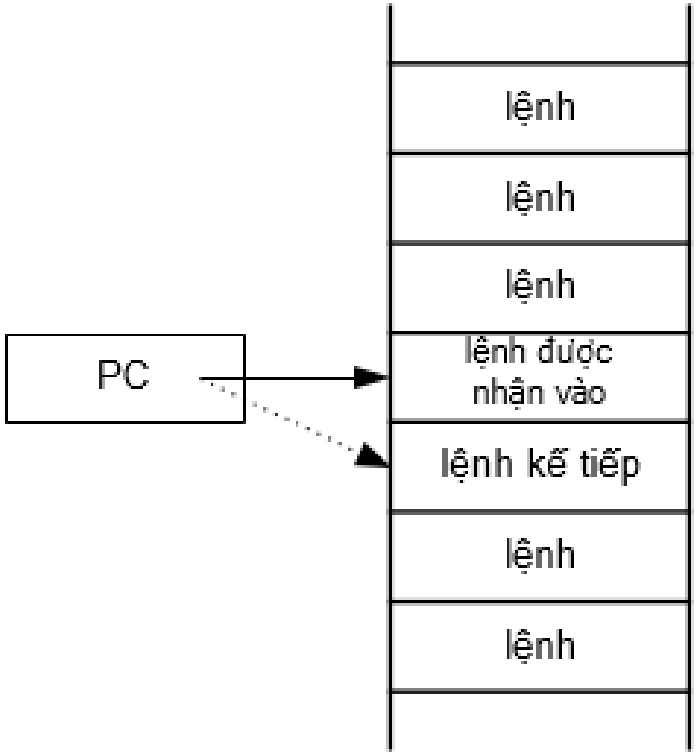
- Kiến trúc tập lệnh (Instruction Set Architecture): cách người lập trình “nhìn" máy tính
- Vi kiến trúc (Microarchitecture): cách thực hiện kiến trúc tập lệnh bằng phần cứng
- Ngôn ngữ trong máy tính:
  - Ngôn ngữ máy (machine language):
    - còn gọi là mã máy (machine code)
    - dạng lệnh máy tính có thể đọc được
    - biểu diễn bằng các bit 0 và 1
  - Hợp ngữ (assembly language):
    - dạng lệnh có thể đọc được bởi con người
    - biểu diễn dạng text

Mô hình lập trình của máy tính



CPU nhận lệnh từ bộ nhớ

- Bộ đếm chương trình PC (Program Counter) là thanh ghi của CPU giữ địa chỉ của lệnh cần nhận vào để thực hiện
- CPU phát địa chỉ từ PC đến bộ nhớ, lệnh được nhận vào
- Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp
- PC tăng bao nhiêu?
  - Tùy thuộc vào độ dài của lệnh vừa được nhận
  - 8086: lệnh có độ dài 8-bit, PC tăng 1



Giải mã và thực hiện lệnh

- Bộ xử lý giải mã lệnh đã được nhận và phát các tín hiệu điều khiển thực hiện thao tác mà lệnh yêu cầu
- Các kiểu thao tác chính của lệnh:
  - Trao đổi dữ liệu giữa CPU với bộ nhớ chính hoặc với cổng vào-ra
  - Thực hiện các phép toán số học hoặc phép toán logic với các dữ liệu (được thực hiện bởi ALU)
  - Chuyển điều khiển trong chương trình (rẽ nhánh, nhảy)

CPU đọc/ghi dữ liệu bộ nhớ

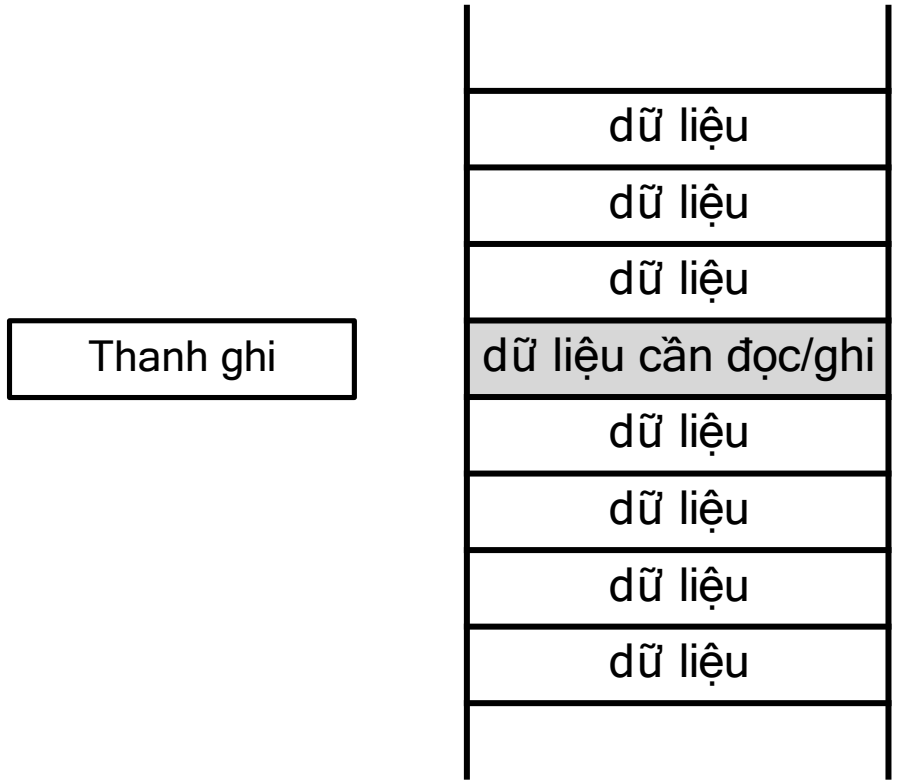
- Với các lệnh trao đổi dữ liệu với bộ nhớ, CPU cần biết và phát ra địa chỉ của ngăn nhớ cần đọc/ghi
- Địa chỉ đó có thể là:
  - Hằng số địa chỉ được cho trực tiếp trong lệnh
  - Giá trị địa chỉ nằm trong thanh ghi con trỏ
  - Địa chỉ = Địa chỉ cơ sở + giá trị dịch chuyển

Hằng số địa chỉ

- Trong lệnh cho hằng số địa chỉ cụ thể
- CPU phát giá trị địa chỉ này đến bộ nhớ để tìm ra ngăn nhớ dữ liệu cần đọc/ghi

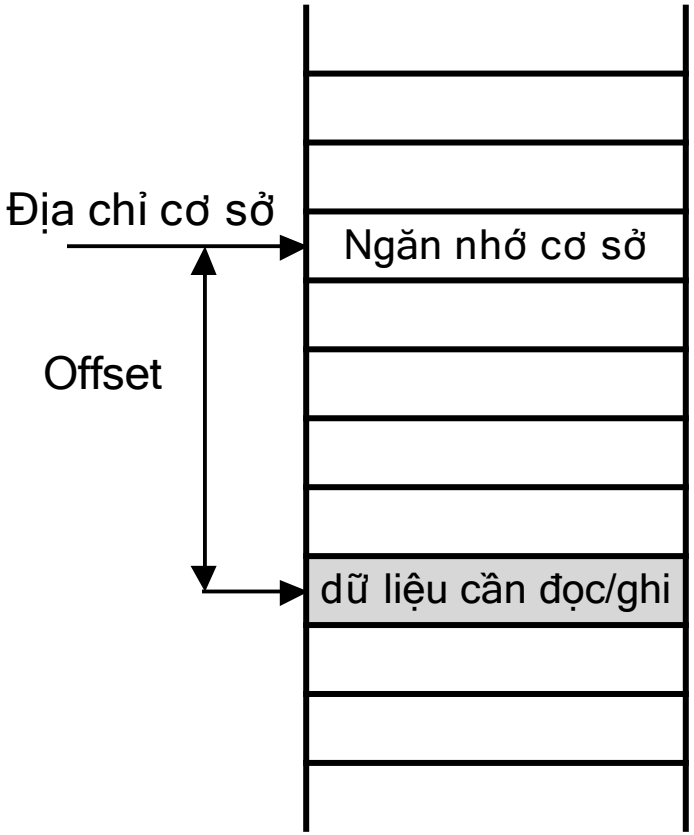
Sử dụng thanh ghi con trỏ

- Trong lệnh cho biết tên thanh ghi con trỏ
- Thanh ghi con trỏ chứa giá trị địa chỉ
- CPU phát địa chỉ này ra để tìm ra ngăn nhớ dữ liệu cần đọc/ghi



Sử dụng địa chỉ cơ sở và dịch chuyển

- Địa chỉ cơ sở (base address): địa chỉ của ngăn nhớ cơ sở
- Giá trị dịch chuyển địa chỉ (offset): gia số địa chỉ giữa ngăn nhớ cần đọc/ghi so với ngăn nhớ cơ sở
- Địa chỉ của ngăn nhớ cần đọc/ghi = (địa chỉ cơ sở) + (offset)
- Có thể sử dụng các thanh ghi để quản lý các tham số này
- Trường hợp riêng:
  - Địa chỉ cơ sở = 0
  - Offset = 0

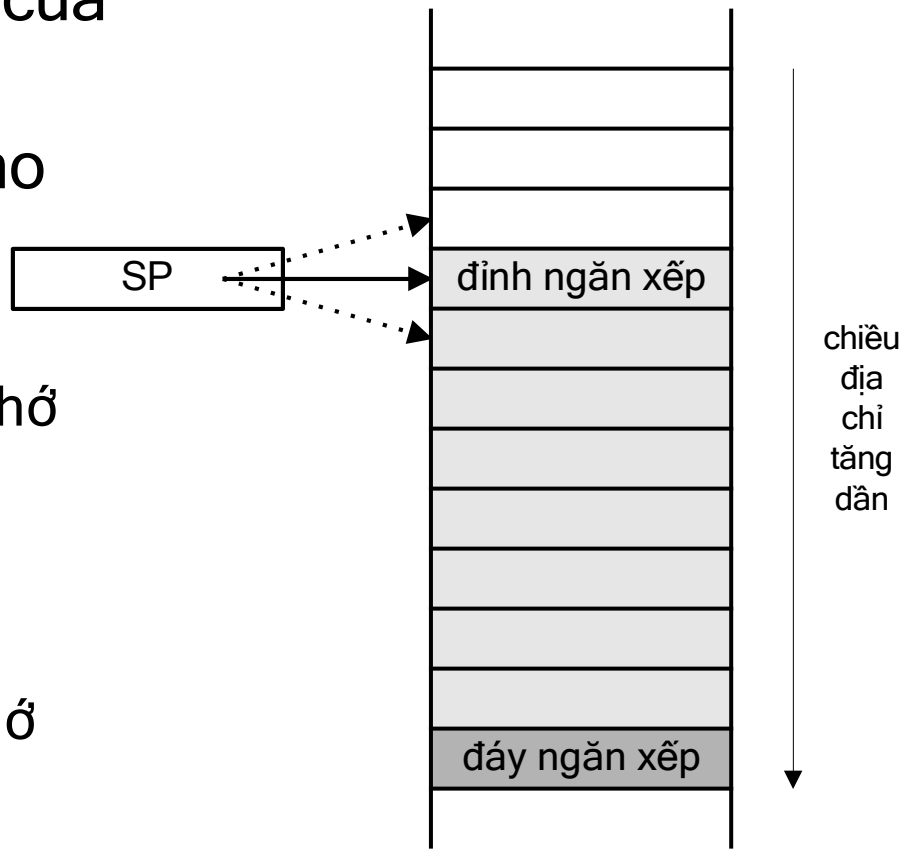


Ngăn xếp (Stack)

- Ngăn xếp là vùng nhớ dữ liệu có cấu trúc LIFO (Last In - First Out vào sau - ra trước)
- Ngăn xếp thường dùng để phục vụ cho chương trình con
- Đáy ngăn xếp là một ngăn 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

Con trỏ ngăn xếp SP (Stack Pointer)

- SP là thanh ghi chứa địa chỉ của ngăn nhớ đỉnh ngăn xếp
- Khi cất thêm một thông tin vào ngăn xếp:
  - Giảm nội dung của SP
  - Thông tin được cất vào ngăn nhớ được trỏ bởi SP
- Khi lấy một thông tin ra khỏi ngăn xếp:
  - Thông tin được đọc từ ngăn nhớ được trỏ bởi SP
  - Tăng nội dung của SP
- Khi ngăn xếp rỗng, SP trỏ vào đáy



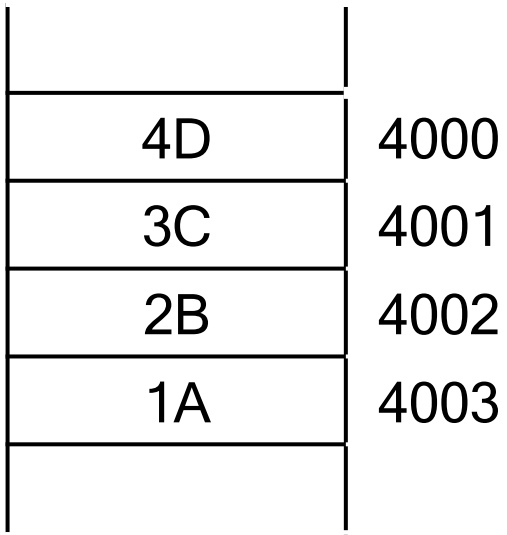
Thứ tự lưu trữ các byte trong bộ nhớ chính

- Bộ nhớ chính được đánh địa chỉ cho từng byte
- Hai cách lưu trữ thông tin nhiều byte:
  - Đầu nhỏ (*Little-endian*): Byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ lớn.
  - Đầu to (*Big-endian*): Byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ lớn.
- Các sản phẩm thực tế:
  - Intel x86: little-endian
  - Motorola 680x0, SunSPARC: big-endian
  - MIPS, IA-64: bi-endian (cả hai kiểu)

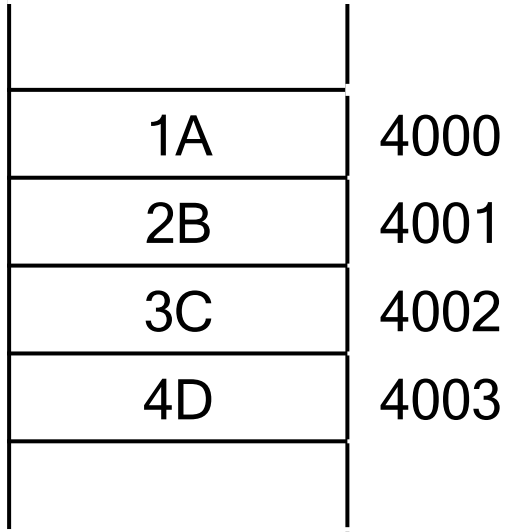
Ví dụ lưu trữ dữ liệu 32-bit

Số nhị phân	0001	1010	0010	1011	0011	1100	0100	1101

Số Hexa	1A	2B	3C	4D



little-endian



big-endian

3.4. Tập lệnh của 8086

3.4.1. Giới thiệu về 8086

- Là bộ vi xử lý 16-bit của Intel, ra mắt năm 1978.
- Mở đầu cho kiến trúc x86 mà chúng ta vẫn sử dụng ngày nay.
- Sử dụng kiến trúc phân đoạn bộ nhớ (segmented memory architecture).
- Có 16 thanh ghi tổng quát 16-bit và các thanh ghi đoạn, thanh ghi con trỏ/chỉ số.

3.4. Tập lệnh của 8086 (tiếp)

3.4.2. Các nhóm lệnh cơ bản

a. Lệnh truyền dữ liệu (Data Transfer Instructions)

- MOV (Move): Sao chép dữ liệu từ nguồn đến đích.

```
MOV AX, 1234h        ;chuyển giá trị hằng 1234h vào thanh ghi AX
MOV BX, CX           ;chuyển nội dung của CX vào BX
MOV AL, [SI]         ;chuyển nội dung ô nhớ được trỏ bởi SI vào thanh ghi AL
```

- PUSH/POP :Đẩy dữ liệu vào stack và lấy dữ liệu ra khỏi stack.

```
PUSH AX ;đẩy nội dung AX vào stack
POP BX  ;lấy nội dung từ stack vào BX
```

- XCHG (Exchange): Hoán đổi nội dung của hai toán hạng.

```
XCHG AX, BX ;hoán đổi nội dung AX và BX
```

3.4. Tập lệnh của 8086 (tiếp)

3.4.3. Các lệnh số học

- ADD (Add): Cộng hai toán hạng. ADD dest, src

```
ADD AX, BX      ;AX = AX + BX
```

- SUB (Subtract): Trừ hai toán hạng. SUB dest, src

```
SUB CX, DX      ;CX = CX - DX
```

- MUL (Multiply - Unsigned): Nhân không dấu. MUL src

```
MUL BL          ;AX = AL * BL
MUL BX          ;DX:AX = AX * BX
```

- DIV (Divide - Unsigned): Chia không dấu.

```
DIV BX          ;AX = AX / BX, DX = AX % BX
```

- INC (Increment): Tăng một đơn vị.

```
INC CX          ;CX = CX + 1
```

- DEC (Decrement): Giảm một đơn vị.

```
DEC DX          ;DX = DX - 1
```

3.4. Tập lệnh của 8086 (tiếp)

3.4.4. Các lệnh logic và bit

- AND (Logical AND): Phép toán AND bitwise.

```
AND AX, 00FFh
```

- OR (Logical OR): Phép toán OR bitwise.

```
OR BX, 1000h
```

- NOT (Logical NOT): Đảo bit (phép bù 1)

```
NOT CX
```

- XOR (Logical XOR): Phép toán XOR bitwise.

```
XOR AX, AX ;xoá AX về 0
```

- SHL (Shift Left): Dịch trái các bit.
- SHR (Shift Right): Dịch phải các bit.
- ROL (Rotate Left): Xoay trái các bit.
- ROR (Rotate Right): Xoay phải các bit.

3.4. Tập lệnh của 8086 (tiếp)

3.4.5. Các lệnh điều khiển luồng

- JMP (Jump): Nhảy không điều kiện đến một nhãn.

  JMP LABEL

- JE (Jump if Equal): Nhảy nếu bằng

  JE LABEL ;nhảy nếu cờ Zero = 1

- JNE (Jump if Not Equal): Nhảy nếu không bằng.

  JNE LABEL ;nhảy nếu cờ Zero = 0

- JL (Jump if Less): Nhảy nếu nhỏ hơn

  JL LABEL

- CALL : Gọi chương trình con, lưu địa chỉ trả về vào stack

  CALL PROC_NAME

- RET : Trở về từ chương trình con, lấy địa chỉ trả về từ stack

  RET

- LOOP: Lặp một khối lệnh một số lần nhất định (dùng CX làm bộ đếm).

  LOOP LABEL

3.4. Tập lệnh của 8086 (tiếp)

3.4.6. Các lệnh xử lý chuỗi

- MOVSB/MOVSW (Move String Byte/Word): Sao chép chuỗi byte/word.
- CMPSB/CMPSW (Compare String Byte/Word): So sánh chuỗi byte/word
- LODSB/LODSW (Load String Byte/Word): Nạp byte/word từ chuỗi vào AL/AX.
- STOSB/STOSW (Store String Byte/Word): Ghi byte/word từ AL/AX vào chuỗi.

3.4.7. Chế độ địa chỉ (Addressing Modes)

1. Chế độ thanh ghi (Register Addressing): Toán hạng là một thanh ghi.

MOV AX, BX

2. Chế độ tức thời (Immediate Addressing): Toán hạng là một giá trị hằng.

MOV AX, 1234h

3. Chế độ trực tiếp (Direct Addressing): Toán hạng là nội dung của một ô nhớ có địa chỉ trực tiếp.

MOV AX, [2000h]

4. Chế độ gián tiếp thanh ghi (Register Indirect Addressing): Địa chỉ của toán hạng được lưu trong một thanh ghi (BX, BP, SI, DI).

MOV AX, [BX]

5. Chế độ cơ sở (Base Addressing): Địa chỉ được tính bằng thanh ghi cơ sở (BX hoặc BP) cộng với một độ dời (displacement).

MOV AL, [BX + 5].

6. Chế độ chỉ số (Index Addressing): Địa chỉ được tính bằng thanh ghi chỉ số (SI hoặc DI) cộng với một độ dời.

MOV AL, [SI + 10]

7. Chế độ cơ sở cộng chỉ số (Base-Index Addressing): Địa chỉ được tính bằng thanh ghi cơ sở cộng thanh ghi chỉ số.

MOV AL, [BX + SI]

8. Chế độ cơ sở cộng chỉ số cộng dịch chuyển (Based Indexed Addressing Mode with Displacement): Địa chỉ được tính bằng thanh ghi cơ sở cộng thanh ghi chỉ số.

MOV AX, [BX + SI + 10] (chuyển nội dung tại địa chỉ BX + SI + 10 vào thanh ghi AX).